### Core documentation

New What's **Release Notes** Bazaar 2.2? what's new in

a detailed log of changes

this release

**User Guide** 

how to use the command line client

**Tutorials** 

brief introductions

**Quick Reference** 

for your wall

**Upgrade Guide** 

moving to Bazaar 2.x

**User Reference** 

all the gory details

System Admin Guide

security, backups, etc.

### Related links

**Desktop Guide** 

how to use our GUI applications

**FAQ** 

frequently asked questions

Glossary

help with terminology

**Developer Docs** 

improving and extending bzr

**Migration Docs** 

for refugees of other tools

Plugins Guide

help on popular plugins

These documents are also available in Spanish, Japanese, and Russian.

♦ Home | Documentation | Table of Contents (2.2b1) »

### What's New in Bazaar 2.2?

This document outlines the major improvements in Bazaar 2.2 vs Bazaar 2.1. As well as summarizing improvements made to the core product, it highlights enhancements within the broader Bazaar world of potential interest to those upgrading.

**Warning:** Bazaar 2.2.0 is currently in development. While less stable than Bazaar 2.1.x, we aim to keep quality at a high level:

- Monthly beta releases should be suitable for early adopters of new features.
- Nightly builds are suitable for users interested in assisting with testing.

If it doubt, please use the latest 2.1 release instead.

# Improved conflict handling

Tree-shape conflicts can be resolved by providing --take-this and --take-other to the bzr resolve command. Just marking the conflict as resolved is still accessible via the --done default action.

# Improved Launchpad integration

Merges can be proposed on Launchpad with the new **Ip-propose-merge** command.

# Notable plugin enhancements

The grep plugin ...

## Better documentation

Numerous improvements have been made to the developer documentation.

# **Enhanced GUI clients**

(Details coming soon.)

### **Further information**

For more detailed information on the changes made, be sure to check the *Bazaar Release Notes* for:

- the interim bzr milestones
- the plugins you use.

For a summary of changes made in earlier releases, see:

• What's New in Bazaar 2.1?

Enjoy, The Bazaar Development Team

Home | Documentation | Table of Contents (2.2b1) »

previous | next

# Introducing Bazaar

#### What is Bazaar?

Bazaar is a tool for helping people collaborate. It tracks the changes that you and other people make to a group of files - such as software source code - to give you snapshots of each stage of their evolution. Using that information, Bazaar can effortlessly merge your work with other people's.

Tools like Bazaar are called version control systems (VCS) and have long been popular with software developers. Bazaar's ease of use, flexibility and simple setup make it ideal not only for software developers but also for other groups who work together on files and documents, such as technical writers, web designers and translators.

This guide takes you through installing Bazaar and how to use it, whether on your own or with a team of other people. If you're already familiar with distributed version control and want to dive straight in, you may wish to skim this section and jump straight to Learning more.

### A brief history of version control systems

Version control tools have been evolving for several decades now. In simple terms, there have been 4 generations of tools:

- 1. file versioning tools, e.g. SCCS, RCS
- 2. tree versioning tools central style, e.g. CVS
- 3. tree versioning tools central style, take two, e.g. Subversion
- 4. tree versioning tools distributed style, e.g. Bazaar.

The design and implementation of Bazaar builds on the lessons learned from all the previous generations of tools. In particular, Bazaar cleanly supports both the central and the distributed version control models so you can change models as it makes sense, without needing to change tools.

#### Central vs distributed VCS

Many traditional VCS tools require a central server which provides the change history or *repository* for a tree of files. To work on the files, users need to connect to the server and *checkout* the files. This gives them a directory or *working tree* in which a person can make changes. To record or *commit* these changes, the user needs access to the central server and they need to ensure they have merged their work with the latest version stored before trying to commit. This approach is known as the centralized model.

The centralized model has proven useful over time but it can have some notable drawbacks. Firstly, a centralized VCS requires that one is able to connect to the server whenever one wants to do version control work. Secondly, the centralized model tightly links the act of **snapshotting** changes with the act of **publishing** those changes. This can be good in some circumstances but it has a negative influence on quality in others.

Distributed VCS tools let users and teams have multiple repositories rather than just a single central one. In Bazaar's case, the history is normally kept in the same place as the code that is being version controlled. This allows the user to commit their changes whenever it makes sense, even when offline. Network access is only required when publishing changes or when accessing changes in another location.

In fact, using distributed VCS tools wisely can have advantages well beyond the obvious one of disconnected operations for developers. Other advantages include:

- easier for developers to create experimental branches
- easier ad-hoc collaboration with peers
- less time required on mechanical tasks more time for

creativity

- increased release management flexibility through the use of "feature-wide" commits
- trunk quality and stability can be kept higher, making everyone's job less stressful
- in open source communities:
  - easier for non-core developers to create and maintain changes
  - easier for core developers to work with non-core developers and move them into the core
- in companies, easier to work with distributed and outsourced teams.

For a detailed look at the advantages of distributed VCS tools over centralized VCS tools, see <a href="http://bazaar-vcs.org/BzrWhy">http://bazaar-vcs.org/BzrWhy</a>.

# Key features of Bazaar

While Bazaar is not the only distributed VCS tool around, it does have some notable features that make it an excellent choice for many teams and communities. A summary of these and comparisons with other VCS tools can be found on the Bazaar Wiki, http://bazaar-vcs.org.

Of the many features, one in particular is worth highlighting: Bazaar is completely free software written in Python. As a result, it is easy to contribute improvements. If you wish to get involved, please see <a href="http://bazaar-vcs.org/BzrSupport">http://bazaar-vcs.org/BzrSupport</a>.

## Learning more

This manual provides an easy to read introduction to Bazaar and how to use it effectively. It is recommended that all users read at least the rest of this chapter as it:

- explains the core concepts users need to know
- introduces some popular ways of using Bazaar to collaborate.

Chapters 2-6 provide a closer look at how to use Bazaar to complete various tasks. It is recommended that most users read these in first-to-last order shortly after starting to use Bazaar. Chapter 7 and beyond provide additional information that helps you make the most of Bazaar once the core functionality is understood. This material can be read when required and in any order.

If you are already familiar with other version control tools, you may wish to get started quickly by reading the following documents:

- Bazaar in five minutes a mini-tutorial
- Bazaar Quick Start Card a one page summary of commonly used commands.

In addition, the online help and Bazaar User Reference provide all the details on the commands and options available.

We hope you find this manual useful. If you have suggestions on how it or the rest of Bazaar's documentation can be improved, please contact us on the mailing list, bazaar@lists.canonical.com.

# Core concepts

## A simple user model

To use Bazaar you need to understand four core concepts:

- **Revision** a snapshot of the files you're working with.
- **Working tree** the directory containing your version-controlled files and sub-directories.
- **Branch** an ordered set of revisions that describe the history of a set of files.
- Repository a store of revisions.

Let's look at each in more detail.

#### Revision

A revision is a *snapshot* of the state of a tree of files and directories, including their content and shape. A revision also has some metadata associated with it, including:

- Who committed it
- When it was committed
- A commit message
- Parent revisions from which it was derived

Revisions are immutable and can be globally, uniquely identified by a *revision-id*. An example revision-id is:

pqm@pqm.ubuntu.com-20071129184101-u9506rihe4zbzyyz

Revision-ids are generated at commit time or, for imports from other systems, at the time of import. While revision-ids are necessary for internal use and external tool integration, branch-specific *revision numbers* are the preferred interface for humans.

Revision numbers are dotted decimal identifiers like 1, 42 and 2977.1.59 that trace a path through the revision number graph for a branch. Revision numbers are generally shorter than revision-ids and, within a single branch, can be compared with each other to get a sense of their relationship. For example, revision 10 is the mainline (see below) revision immediately after revision 9. Revision numbers are generated on the fly when commands are executing, because they depend on which revision is the tip (i.e. most recent revision) in the branch.

See Specifying revisions in the appendices for a closer look at the numerous ways that revisions and ranges of revisions can be specified in Bazaar, and Understanding Revision Numbers for a

more detailed description of revision numbering.

# **Working Tree**

A working tree is a *version-controlled directory* holding files the user can edit. A working tree is associated with a *branch*.

Many commands use the working tree as their context, e.g. commit makes a new revision using the current content of files in the working tree.

#### Branch

In the simplest case, a branch is an *ordered series of revisions*. The last revision is known as the *tip*.

Branches may split apart and be *merged* back together, forming a *graph* of revisions. Technically, the graph shows directed relationships (between parent and child revisions) and there are no loops, so you may hear some people refer to it as a *directed acyclic graph* or DAG.

If this name sounds scary, don't worry. The important things to remember are:

- The primary line of development within the DAG is called the *mainline*, *trunk*, or simply the *left hand side* (LHS).
- A branch might have other lines of development and if it does, these other lines of development begin at some point and end at another point.

# Repository

A repository is simply a *store of revisions*. In the simplest case, each branch has its own repository. In other cases, it makes sense for branches to share a repository in order to optimize disk usage.

### Putting the concepts together

Once you have grasped the concepts above, the various ways of using Bazaar should become easier to understand. The simplest way of using Bazaar is to use a *standalone tree*, which has a working tree, branch, and repository all in a single location. Other common scenarios include:

- Shared repositories working tree and branch are colocated, but the repository is in a higher level directory.
- Stacked branches branch stores just its unique revisions, using its parent's repository for common revisions.
- Lightweight checkouts branch is stored in a different location to the working tree.

The best way to use Bazaar, however, depends on your needs. Let's take a look at some common workflows next.

# Workflows

### Bazaar is just a tool

Bazaar supports many different ways of working together. This means that you can start with one workflow and adapt it over time as circumstances change. There is no "one true way" that always makes sense and there never will be. This section provides a brief overview of some popular workflows supported by Bazaar.

Keep in mind that these workflow are just *some* examples of how Bazaar can be used. You may want to use a workflow not listed here, perhaps building on the ideas below.

#### Solo

Whether developing software, editing documents or changing configuration files, having an easy-to-use VCS tool can help. A single user can use this workflow effectively for managing projects where they are the only contributor.

- ① create project
- record changes
- **3** browse history
- package release



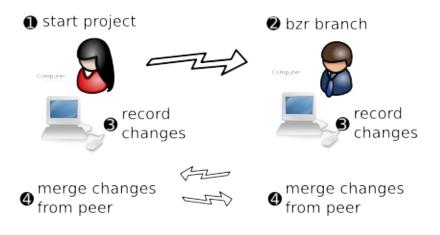
Advantages of this workflow over not using version control at all include:

- backup of old versions
- rollback to an earlier state
- tracking of history.

The key features of Bazaar appropriate for this workflow are low administration (no server setup) and ease of use.

#### Partner

Sometimes two people need to work together sharing changes as they go. This commonly starts off as a *Solo* workflow (see above) or a team-oriented workflow (see below). At some point, the second person takes a branch (copy including history) of what the first person has done. They can then work in parallel exchanging changes by merging when appropriate.



#### Advantages over Solo are:

- easier sharing of changes
- each line of each text file can be attributed to a particular change including who changed it, when and why.

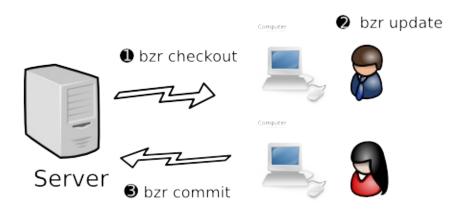
When implementing this workflow, Bazaar's advantages over CVS and Subversion include:

no server to setup

• intelligent merging means merging multiple times isn't painful.

#### Centralized

Also known as *lock-step*, this is essentially the same as the workflow encouraged/enforced by CVS and Subversion. All developers work on the same branch (or branches). They run bzr update to get their checkout up-to-date, then bzr commit to publish their changes to the central location.

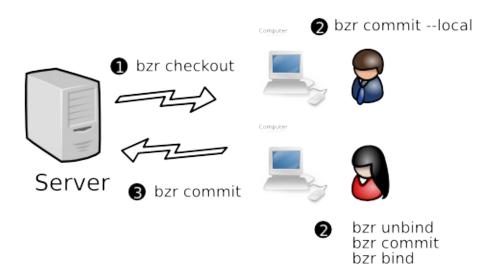


Subversion and CVS are good choices for implementing this workflow because they make it easy. Bazaar directly supports it as well while providing some important advantages over CVS and Subversion:

- better branching and merging
- better renaming support.

#### Centralized with local commits

This is essentially the same as the *Centralized* model, except that when developers are making a series of changes, they do commit --local or unbind their checkout. When it is complete, they commit their work to the shared mainline.



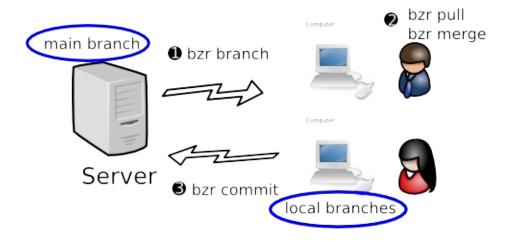
#### Advantages over Centralized:

- Can work offline, e.g. when disconnected during travel
- Less chance for a bad commit to interfere with everyone else's work

Subversion and CVS do not support this model. Other distributed VCS tools can support it but do so less directly than Bazaar does.

#### Decentralized with shared mainline

In this workflow, each developer has their own branch or branches, plus commit rights to the main branch. They do their work in their personal branch, then merge it into the mainline when it is ready.



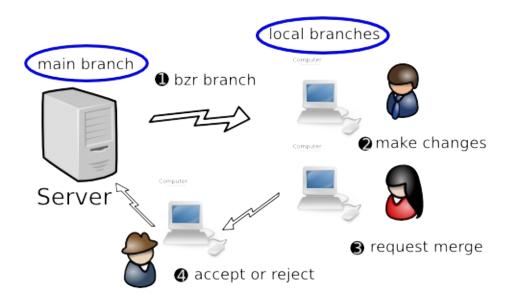
Advantage over Centralized with local commits:

- Easier organization of work separate changes can be developed in their own branches
- Developers can merge one another's personal branches when working on something together.

Subversion and CVS do not support this model. Other distributed VCS tools support it. Many features of Bazaar are good for this workflow including ease of use, shared repositories, integrated merging and rich metadata (including directory rename tracking).

### Decentralized with human gatekeeper

In this workflow, each developer has their own branch or branches, plus read-only access to the main branch. One developer (the gatekeeper) has commit rights to the main branch. When a developer wants their work merged, they ask the gatekeeper to merge it. The gatekeeper does code review, and merges the work into the main branch if it meets the necessary standards.



Advantage over Decentralized with shared mainline:

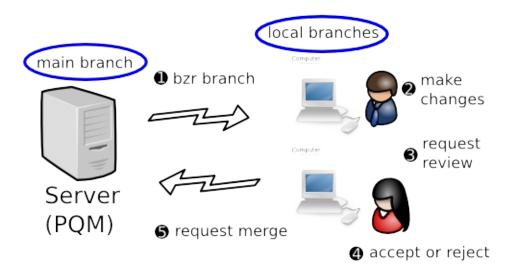
- Code is always reviewed before it enters the mainline
- Tighter control over when changes get incorporated into the mainline.

A companion tool of Bazaar's called Bundle Buggy can be very useful for tracking what changes are up for review, their status and reviewer comments.

### Decentralized with automatic gatekeeper

In this workflow, each developer has their own branch or branches, plus read-only access to the mainline. A software gatekeeper has commit rights to the main branch. When a developer wants their work merged, they request another person to review it. Once it has passed review, either the original author or the reviewer asks the gatekeeper software to merge it, depending on team policies. The gatekeeper software does a merge, a compile, and runs the test suite. If and only if the code passes, it is merged into the mainline.

Note: As an alternative, the review step can be skipped and the author can submit the change to the automatic gatekeeper without it. (This is particularly appropriate when using practices such as Pair Programming that effectively promote just-in-time reviews instead of reviewing code as a separate step.)



Advantages over Decentralized with human gatekeeper:

- Code is always tested before it enters the mainline (so the integrity of the mainline is higher)
- Scales better as teams grow.

A companion tool of Bazaar's called Patch Queue Manager (PQM) can provide the automated gatekeeper capability.

### Implementing a workflow

For an in-depth look at how to implement each of the workflows above, see chapters 3 to 6 in this manual. First though, chapter 2 explains some important pre-requisites including installation, general usage instructions and configuration tips.

Home | Documentation | Table of Contents (2.2b1) » Bazaar User Guide » previous | next

## Installing Bazaar

### Linux

Bazaar packages are available for most popular Linux distributions including Ubuntu/Debian, Red Hat and Gentoo. See <a href="http://bazaar-vcs.org/Download">http://bazaar-vcs.org/Download</a> for the latest instructions.

### Windows

For Windows users, an installer is available that includes the core Bazaar package together with necessary pre-requisites and some useful plug-ins. See <a href="http://bazaar-vcs.org/Download">http://bazaar-vcs.org/Download</a> for the latest instructions.

Note: If you are running Cygwin on Windows, a Bazaar for Cygwin package is available and ought to be used instead of the Windows version.

## Other operating systems

Beyond Linux and Windows, Bazaar packages are available for a large range of other operating systems include Mac OS X, FreeBSD and Solaris. See <a href="http://bazaar-vcs.org/Download">http://bazaar-vcs.org/Download</a> for the latest instructions.

### Installing from scratch

If you wish to install Bazaar from scratch rather than using a pre-built package, the steps are:

- 1. If it is not installed already, install Python 2.4 or later.
- 2. Download the bazaar-xxx.tar.gz file (where xxx is the version number) from http://bazaar-vcs.org/Download or from Launchpad (https://launchpad.net/~bzr/).
- 3. Unpack the archive using tar, WinZip or equivalent.
- 4. Put the created directory on your PATH.

To test the installation, try running the **bzr** command like this:

```
bzr version
```

This will display the version of Bazaar you have installed. If this doesn't work, please contact us via email or IRC so we can help you get things working.

#### Installing into site-wide locations

Instead of adding the directory to your PATH, you can install bzr into the system locations using:

```
python setup.py install
```

If you do not have a compiler, or do not have the python development tools installed, bzr supplies a (slower) pure-python implementation of all extensions. You can install without compiling extensions with:

python setup.py install build\_ext --allow-python-fallback

### Running the development version

You may wish to always be using the very latest development version of Bazaar. Note that this is not recommended for the majority of users as there is an increased risk of bugs. On the other hand, the development version is remarkably solid (thanks to the processes we follow) and running it makes it easier for you to send us changes for bugs and improvements. It also helps us by having more people testing the latest software.

Here are the steps to follow:

- 1. Install Bazaar using one of the methods given above.
- 2. Get a copy of the development version like this:

```
bzr branch lp:bzr
```

3. Put the created directory on your PATH.

Advanced users may also wish to build the optional C extensions for greater speed. This can be done using make and requires pyrex and a C compiler. Please contact us on email or IRC if you need assistance with this.

### Running multiple versions

It's easy to have multiple versions of Bazaar installed and to switch between them. To do this, simply provide the full pathname to the **bzr** command you wish to run. The relevant libraries will be automatically detected and used. Of course, if you do not provide a pathname, then the **bzr** used will be the one found on your system path as normal.

Note that this capability is particularly useful if you wish to run (or test) both the latest released version and the development version say.

Home | Documentation | Table of Contents (2.2b1) » Bazaar User Guide » previous | next

# **Entering commands**

### User interfaces

There are numerous user interfaces available for Bazaar. The core package provides a command line tool called **bzr** and graphical user interfaces (GUIs) are available as plug-ins.

### Using bzr

#### The syntax is:

bzr [global-options] command [options and arguments]

Global options affect how Bazaar operates and can appear either before or after command. Command specific options must appear after the command but may be given either before, during or after any command-specific arguments.

### Common options

Some options are legal for all commands as shown below.

Short form	Long form	Description
-h	–help	get help
-V	-verbose	be more verbose
-q	–quiet	be more quiet

Quiet mode implies that only errors and warnings are displayed. This can be useful in scripts.

Note: Most commands typically only support one level of verbosity though that may change in the future. To ask for a higher level of verbosity, simply specify the -v option multiple times.

## Getting help

Bazaar comes with a built-in on-line help system, accessed through:

```
bzr help
```

You can ask for help on a command, or on non-command topics. To see a list of available help of each kind, use either:

```
bzr help commands
bzr help topics
```

For help on a particular command, use either of these forms:

```
bzr help status
bzr status --help
```

If you wish to search the help or read it as a larger document, the information is also available in the Bazaar User Reference.

Home | Documentation | Table of Contents (2.2b1) » Bazaar User Guide » previous | next

# **Configuring Bazaar**

### Telling Bazaar about yourself

One function of a version control system is to keep track of who changed what. In a decentralized system, that requires an identifier for each author that is globally unique. Most people already have one of these: an email address. Bazaar is smart enough to automatically generate an email address by looking up your username and hostname. If you don't like the guess that Bazaar makes, then use the whoami command to set the identifier you want:

% bzr whoami "Your Name <email@example.com>"

If whoami is used without an argument, the current value is displayed.

### Configuration files

Configuration files are located in \$HOME/.bazaar on Linux/Unix and C:\Documents and Settings\<username>\Application Data\Bazaar\2.0 on Windows. There are three primary configuration files in this location:

- bazaar.conf describes default configuration options,
- locations.conf describes configuration information for specific branch locations,
- authentication.conf describes credential information for remote servers.

Each branch can also contain a configuration file that sets values specific to that branch. This file is found at <code>.bzr/branch/branch.conf</code> within the branch. This file is visible to **all users of a branch**. If you wish to override one of the values for a branch with a setting that is specific to you, then you can do so in <code>locations.conf</code>.

Here is sample content of bazaar.conf after setting an email address using the whoami command:

```
[DEFAULT]
email = Your Name <email@example.com>
```

For further details on the syntax and configuration settings supported, see Configuration Settings in the Bazaar User Reference.

### Rule-based preferences

Some commands and plugins provide custom processing on files matching certain patterns. Per-user rule-based preferences are defined in BZR\_HOME/rules.

For further information on how rules are searched and the detailed syntax of the relevant files, see Rules in the Bazaar User Reference.

Home | Documentation | Table of Contents (2.2b1) » Bazaar User Guide » previous | next

# Using aliases

### What are aliases?

Aliases are an easy way to create shortcuts for commonly-typed commands, or to set defaults for commands.

### Defining aliases

Command aliases can be defined in the [ALIASES] section of your bazaar.conf file. Aliases start with the alias name, then an equal sign, then a command fragment. Here's an example ALIASES section:

```
[ALIASES]
recentlog=log -r-3..-1
ll=log --line -r-10..-1
commit=commit --strict
diff=diff --diff-options -p
```

Here are the explanations of the examples above:

- The first alias makes a new recentlog command that shows the logs for the last three revisions
- The 11 alias shows the last 10 log entries in line format.
- the commit alias sets the default for commit to refuse to commit if new files in the tree are not recognized.
- the diff alias adds the coveted -p option to diff

## Using the aliases

The aliases defined above would be used like so:

```
% bzr recentlog
% bzr ll
% bzr commit
% bzr diff
```

### Rules for aliases

- You can override a portion of the options given in an alias by specifying the new part on the command-line. For example, if you run lastlog -r-5..., you will only get five line-based log entries instead of 10. Note that all boolean options have an implicit inverse, so you can override the commit alias with commit --no-strict.
- Aliases can override the standard behaviour of existing commands by giving an alias name that is the same as the original command. For example, default commit is changed with commit --strict.
- Aliases cannot refer to other aliases. In other words making a lastlog alias and referring to it with a 11 alias will not work. This includes aliases that override standard commands.
- Giving the --no-aliases option to the bzr command will tell it to ignore aliases for that run. For example, running bzr --no-aliases commit will perform a standard commit instead, not do a commit --strict.

# Using plugins

### What is a plugin?

A plugin is an external component for Bazaar that is typically made by third parties. A plugin is capable of augmenting Bazaar by adding new functionality. A plugin can also change current Bazaar behavior by replacing current functionality. Sample applications of plugins are:

- overriding commands
- adding new commands
- providing additional network transports
- customizing log output.

The sky is the limit for the customization that can be done through plugins. In fact, plugins often work as a way for developers to test new features for Bazaar prior to inclusion in the official codebase. Plugins are helpful at feature retirement time as well, e.g. deprecated file formats may one day be removed from the Bazaar core and be made available as a plugin instead.

Plugins are good for users, good for external developers and good for Bazaar itself.

## Where to find plugins

We keep our list of plugins on the http://bazaar-vcs.org/BzrPlugins page.

### How to install a plugin

Installing a plugin is very easy! If not already created, create a directory under your Bazaar configuration plugins directory, Linux and ~/.bazaar/ on C:\Documents and Settings\ <username>\Application Data\Bazaar\2.0\ on Windows. Within this directory (referred to as \$BZR HOME below), each plugin is placed in its own subdirectory.

Plugins work particularly well with Bazaar branches. For example, to install the bzrtools plugins for your main user account on Linux, one can perform the following:

```
bzr branch http://panoramicfeedback.com/opensource/bzr/bzrtools
~/.bazaar/plugins/bzrtools
```

When installing plugins, the directories that you install them in must be valid python identifiers. This means that they can only contain certain characters, notably they cannot contain hyphens (-). Rather than installing bzr-gtk to \$BZR\_HOME/plugins/bzr-gtk, install it to \$BZR\_HOME/plugins/gtk.

# Alternative plugin locations

If you have the necessary permissions, plugins can also be installed on a system-wide basis. One can additionally override the personal plugins location by setting the environment variable BZR\_PLUGIN\_PATH (see User Reference for a detailed explanation).

# Listing the installed plugins

To do this, use the plugins command like this:

#### bzr plugins

The name, location and version of each plugin installed will be displayed.

New commands added by plugins can be seen by running bzr help commands. The commands provided by a plugin are shown followed by the name of the plugin in brackets.

# Popular plugins

Here is a sample of some of the more popular plugins.

Name	Description
QBzr	Qt-based GUI tools
bzr-gtk	GTK-based GUI tools
bzr-eclipse	Eclipse integration
bzrtools	misc. enhancements including shelf
difftools	external diff tool helper
extmerge	external merge tool helper
bzr-svn	use Subversion as a repository
cvsps	migrate CVS patch-sets
	QBzr bzr-gtk bzr-eclipse bzrtools difftools extmerge bzr-svn

If you wish to write your own plugins, it is not difficult to do. See Writing a plugin in the appendices to get started.

# Bazaar Zen

# **Grokking Bazaar**

While Bazaar is similar to other VCS tools in many ways, there are some important differences that are not necessarily obvious at first glance. This section attempts to explain some of the things users need to know in order to "grok" Bazaar, i.e. to deeply understand it.

Note: It isn't necessary to fully understand this section to use Bazaar. You may wish to skim this section now and come back to it at a later time.

# Understanding revision numbers

All revisions in the mainline of a branch have a simple increasing integer. (First commit gets 1, 10th commit gets 10, etc.) This makes them fairly natural to use when you want to say "grab the 10th revision from my branch", or "fixed in revision 3050".

For revisions which have been merged into a branch, a dotted notation is used (e.g., 3112.1.5). Dotted revision numbers have three numbers [1]. The first number indicates what mainline revision change is derived from. The second number is the branch counter. There can be many branches derived from the same revision, so they all get a unique number. The third number is the number of revisions since the branch started. For example, 3112.1.5 is the first branch from revision 3112, the fifth revision on that branch.

Versions prior to bzr 1.2 used a slightly different algorithm.

[1] Some nested branches would get extra numbers (such as 1.1.1.1.1) rather than the simpler 3-number system.

# Hierarchical history is good

Imagine a project with multiple developers contributing changes where many changes consist of a series of commits. To give a concrete example, consider the case where:

- The tip of the project's trunk is revision 100.
- Mary makes 3 changes to deliver feature X.
- Bill makes 4 changes to deliver feature Y.

If the developers are working in parallel and using a traditional centralized VCS approach, the project history will most likely be linear with Mary's changes and Bill's changes interleaved. It might look like this:

```
107: Add documentation for Y
106: Fix bug found in testing Y
105: Fix bug found in testing X
104: Add code for Y
103: Add documentation for X
102: Add code and tests for X
101: Add tests for Y
100: ...
```

Many teams use this approach because their tools make branching and merging difficult. As a consequence, developers update from and commit to the trunk frequently, minimizing integration pain by spreading it over every commit. If you wish, you can use Bazaar exactly like this. Bazaar does offer other ways though that you ought to consider.

An alternative approach encouraged by distributed VCS tools is to create feature branches and to integrate those when they are ready. In this case, Mary's feature branch would look like this:

```
103: Fix bug found in testing X
```

```
102: Add documentation for X
101: Add code and tests for X
100: ...
```

#### And Bill's would look like this:

```
104: Add documentation for Y
103: Fix bug found in testing Y
102: Add code for Y
101: Add tests for Y
100: ...
```

If the features were independent and you wanted to keep linear history, the changes could be pushed back into the trunk in batches. (Technically, there are several ways of doing that but that's beyond the scope of this discussion.) The resulting history might look like this:

```
107: Fix bug found in testing X
106: Add documentation for X
105: Add code and tests for X
104: Add documentation for Y
103: Fix bug found in testing Y
102: Add code for Y
101: Add tests for Y
100: ...
```

While this takes a bit more effort to achieve, it has some advantages over having revisions randomly intermixed. Better still though, branches can be merged together forming a non-linear history. The result might look like this:

```
102: Merge feature X
            100.2.3: Fix bug found in testing X
            100.2.2: Add documentation for X
            100.2.1: Add code and tests for X

101: Merge feature Y
            100.1.4: Add documentation for Y
            100.1.3: Fix bug found in testing Y
            100.1.2: Add code for Y
            100.1.1: Add tests for Y
```

```
100: ...
```

#### Or more likely this:

```
102: Merge feature X
        100.2.3: Fix bug
        100.2.2: Add documentation
        100.2.1: Add code and tests

101: Merge feature Y
        100.1.4: Add documentation
        100.1.3: Fix bug found in testing
        100.1.2: Add code
        100.1.1: Add tests
```

This is considered good for many reasons:

- It makes it easier to understand the history of a project. Related changes are clustered together and clearly partitioned.
- You can easily collapse history to see just the commits on the mainline of a branch. When viewing the trunk history like this, you only see high level commits (instead of a large number of commits uninteresting at this level).
- If required, it makes backing out a feature much easier.
- Continuous integration tools can be used to ensure that all tests still pass before committing a merge to the mainline. (In many cases, it isn't appropriate to trigger CI tools after every single commit as some tests will fail during development. In fact, adding the tests first - TDD style - will guarantee it!)

In summary, the important points are:

Organize your work using branches.

Integrate changes using merge.

Ordered revision numbers and hierarchy make history easier to follow.

# Each branch has its own view of history

As explained above, Bazaar makes the distinction between:

- mainline revisions, i.e. ones you committed in your branch, and
- merged revisions, i.e. ones added as ancestors by committing a merge.

Each branch effectively has its own view of history, i.e. different branches can give the same revision a different "local" revision number. Mainline revisions always get allocated single number revision numbers while merged revisions always get allocated dotted revision numbers.

To extend the example above, here's what the revision history of Mary's branch would look like had she decided to merge the project trunk into her branch after completing her changes:

Once again, it's easy for Mary to look at just *her* top level of history to see the steps she has taken to develop this change. In this context, merging the trunk (and resolving any conflicts caused by doing that) is just one step as far as the history of this branch is concerned.

It's important to remember that Bazaar is not changing history here, nor is it changing the global revision identifiers. You can always use the latter if you really want to. In fact, you can use the branch specific revision numbers when communicating as long as you provide the branch URL as context. (In many Bazaar projects, developers imply the central trunk branch if they exchange a revision number without a branch URL.)

Merges do not change revision numbers in a branch, though they do allocate local revision numbers to newly merged revisions. The only time Bazaar will change revision numbers in a branch is when you explicitly ask it to mirror another branch.

Note: Revisions are numbered in a stable way: if two branches have the same revision in their mainline, all revisions in the ancestry of that revision will have the same revision numbers. For example, if Alice and Bob's branches agree on revision 10, they will agree on all revisions before that.

# Summary

In general, if you follow the advice given earlier - organise your work in branches and use merge to collaborate - you'll find Bazaar generally does what you expect.

In coming chapters, we examine various ways of using Bazaar beginning with the simplest: using Bazaar for personal projects.

Home | Documentation | Table of Contents (2.2b1) » Bazaar User Guide » previous | next

# Going solo

# A personal productivity tool

Some tools are designed to make individuals productive (e.g. editors) while other tools (e.g. back-end services) are focused on making teams or whole companies more productive. Version control tools have traditionally been in the latter camp.

One of the cool things about Bazaar is that it is so easy to setup that version control can now be treated as a personal productivity tool. If you wish to record changes to files for the purposes of checkpointing good known states or tracking history, it is now easy to do so. This chapter explains how.

## The solo workflow

If you are creating your own masterpiece, whether that be a software project or a set of related documents, the typical workflow looks like this:

- ① create project
- 2 record changes
- 3 browse history
- package release



Even if you will always be working as part of a team, the tasks covered in this chapter will be the basis of what you'll be doing so it's a good place to start.

Home | Documentation | Table of Contents (2.2b1) » Bazaar User Guide » previous | next

# Starting a project

# Putting an existing project under version control

If you already have a tree of source code (or directory of documents) you wish to put under version control, here are the commands to use:

```
cd my-stuff
bzr init
bzr add
bzr commit -m "Initial import"
```

bzr init creates a .bzr directory in the top level directory (my-stuff in the example above). Note that:

- Bazaar has everything it needs in that directory you do not need to setup a database, web server or special service to use it
- Bazaar is polite enough to only create one .bzr in the directory given, not one in every subdirectory thereof.

bzr add then finds all the files and directories it thinks ought to be version controlled and registers them internally. bzr commit then records a snapshot of the content of these and records that information, together with a commit message.

More information on init, add and commit will be provided later. For now, the important thing to remember is the recipe above.

# Starting a new project

If you are starting a project from scratch, you can also use the recipe above, after creating an empty directory first of course. For efficiency reasons that will be explored more in later chapters though, it is a good idea to create a repository for the project at the top level and to nest a *main* branch within it like this:

```
bzr init-repo my.repo
cd my.repo
bzr init my.main
cd my.main
hack, hack, hack
bzr add
bzr commit -m "Initial import"
```

Some users prefer a name like *trunk* or *dev* to *main*. Choose whichever name makes the most sense to you.

Note that the init-repo and init commands both take a path as an argument and will create that path if it doesn't already exist.

# Controlling file registration

#### What does Bazaar track?

As explained earlier, bzr add finds and registers all the things in and under the current directory that Bazaar thinks ought to be version controlled. These things may be:

- files
- directories
- symbolic links.

Bazaar has default rules for deciding which files are interesting and which ones are not. You can tune those rules as explained in Ignoring files below.

Unlike many other VCS tools, Bazaar tracks directories as first class items. As a consequence, empty directories are correctly supported - you don't need to create a dummy file inside a directory just to ensure it gets tracked and included in project exports.

For symbolic links, the value of the symbolic link is tracked, not the content of the thing the symbolic link is pointing to.

Note: Support for tracking projects-within-projects ("nested trees") is currently under development. Please contact the Bazaar developers if you are interested in helping develop or test this functionality.

# Selective registration

In some cases, you may want or need to explicitly nominate the things to register rather than leave it up to Bazaar to find things. To do this, simply provide paths as arguments to the add command like this:

bzr add fileX dirY/

Adding a directory implicitly adds all interesting things underneath it.

# Ignoring files

Many source trees contain some files that do not need to be versioned, such as editor backups, object or bytecode files, and built programs. You can simply not add them, but then they'll always crop up as unknown files. You can also tell Bazaar to ignore these files by adding them to a file called .bzrignore at the top of the tree.

This file contains a list of file wildcards (or "globs"), one per line. Typical contents are like this:

```
*.o
*~
*.tmp
*.py[co]
```

If a glob contains a slash, it is matched against the whole path from the top of the tree; otherwise it is matched against only the filename. So the previous example ignores files with extension .o in all subdirectories, but this example ignores only config.h at the top level and HTML files in doc/:

```
./config.h
doc/*.html
```

To get a list of which files are ignored and what pattern they matched, use bzr ignored:

```
% bzr ignored config.h ./config.h configure.in~ *~
```

Note that ignore patterns are only matched against non-versioned files, and control whether they are treated as "unknown" or "ignored". If a file is explicitly added, it remains versioned regardless of whether

it matches an ignore pattern.

The .bzrignore file should normally be versioned, so that new copies of the branch see the same patterns:

```
% bzr add .bzrignore
% bzr commit -m "Add ignore patterns"
```

The command bzr ignore PATTERN can be used to easily add PATTERN to the .bzrignore file (creating it if necessary and registering it to be tracked by Bazaar). Removing and modifying patterns are done by directly editing the .bzrignore file.

# Global ignores

There are some ignored files which are not project specific, but more user specific. Things like editor temporary files, or personal temporary files. Rather than add these ignores to every project, bzr supports a global ignore file in ~/.bazaar/ignore [1]. It has the same syntax as the per-project ignore file.

On Windows, the users configuration files can be found in the application data directory. So instead of ~/.bazaar/branch.conf the configuration file can be found as: C:\Documents and Settings\<username>\Application
Data\Bazaar\2.0\branch.conf. The same is true for locations.conf, ignore, and the plugins directory.

Home | Documentation | Table of Contents (2.2b1) » Bazaar User Guide » previous | next

# Reviewing changes

# Looking before you leap

Once you have completed some work, it's a good idea to review your changes prior to permanently recording it. This way, you can make sure you'll be committing what you intend to.

Two bzr commands are particularly useful here: status and diff.

## bzr status

The **status** command tells you what changes have been made to the working directory since the last revision:

```
% bzr status
modified:
foo
```

bzr status hides "boring" files that are either unchanged or ignored. The status command can optionally be given the name of some files or directories to check.

#### bzr diff

The **diff** command shows the full text of changes to all files as a standard unified diff. This can be piped through many programs such as "patch", "diffstat", "filterdiff" and "colordiff":

```
% bzr diff

=== added file 'hello.txt'

--- hello.txt 1970-01-01 00:00:00 +0000

+++ hello.txt 2005-10-18 14:23:29 +0000

@@ -0,0 +1,1 @@

+hello world
```

With the <u>-r</u> option, the tree is compared to an earlier revision, or the differences between two versions are shown:

To see the changes introduced by a single revision, you can use the -c option to diff.

The --diff-options option causes bzr to run the external diff program, passing options. For example:

```
% bzr diff --diff-options --side-by-side foo
```

Some projects prefer patches to show a prefix at the start of the path for old and new files. The --prefix option can be used to provide such a prefix. As a shortcut, bzr diff -p1 produces a form that works with the command patch -p1.

Home | Documentation | Table of Contents (2.2b1) » Bazaar User Guide » previous | next

## Recording changes

#### bzr commit

When the working tree state is satisfactory, it can be **committed** to the branch, creating a new revision holding a snapshot of that state.

The **commit** command takes a message describing the changes in the revision. It also records your userid, the current time and timezone, and the inventory and contents of the tree. The commit message is specified by the -m or --message option. You can enter a multi-line commit message; in most shells you can enter this just by leaving the quotes open at the end of the line.

```
% bzr commit -m "added my first file"
```

You can also use the **-F** option to take the message from a file. Some people like to make notes for a commit message while they work, then review the diff to make sure they did what they said they did. (This file can also be useful when you pick up your work after a break.)

#### Message from an editor

If you use neither the -m nor the -F option then bzr will open an editor for you to enter a message. The editor to run is controlled by your \$VISUAL or \$EDITOR environment variable, which can be overridden by the editor setting in ~/.bazaar/bazaar.conf; \$BZR\_EDITOR will override either of the above mentioned editor options. If you quit the editor without making any changes, the commit will be cancelled.

The file that is opened in the editor contains a horizontal line. The part of the file below this line is included for information only, and will not form part of the commit message. Below the separator is shown the list of files that are changed in the commit. You should write your message above the line, and then save the file and exit.

If you would like to see the diff that will be committed as you edit the message you can use the --show-diff option to commit. This will include the diff in the editor when it is opened, below the separator and the information about the files that will be committed. This means that you can read it as you write the message, but the diff itself wont be seen in the commit message when you have finished. If you would like parts to be included in the message you can copy and paste them above the separator.

#### Selective commit

If you give file or directory names on the commit command line then only the changes to those files will be committed. For example:

```
% bzr commit -m "documentation fix" commit.py
```

By default bzr always commits all changes to the tree, even if run from a subdirectory. To commit from only the current directory down, use:

```
% bzr commit .
```

#### Giving credit for a change

If you didn't actually write the changes that you are about to commit, for instance if you are applying a patch from someone else, you can use the --author commit option to give them credit for the change:

```
% bzr commit --author "Jane Rey <jrey@example.com>"
```

The person that you specify there will be recorded as the "author" of the revision, and you will be recorded as the "committer" of the revision.

If more than one person works on the changes for a revision, for instance if you are pair-programming, then you can record this by specifying --author multiple times:

```
% bzr commit --author "Jane Rey <jrey@example.com>" \
    --author "John Doe <jdoe@example.com>"
```

Home | Documentation | Table of Contents (2.2b1) » Bazaar User Guide » previous | next

# Browsing history

#### bzr log

The bzr log command shows a list of previous revisions.

As with bzr diff, bzr log supports the -r argument:

```
% bzr log -r 1000.. # Revision 1000 and everything aft bzr log -r ..1000 # Everything up to and including r bzr log -r 1000..1100 # changes from 1000 to 1100 # The changes in only revision 100
```

#### Viewing merged revisions

As distributed VCS tools like Bazaar make merging much easier than it is in central VCS tools, the history of a branch may often contain lines of development splitting off the mainline and merging back in at a later time. Technically, the relationship between the numerous revision nodes is known as a Directed Acyclic Graph or DAG for short.

In many cases, you typically want to see the mainline first and drill down from there. The default behaviour of log is therefore to show the mainline and indicate which revisions have nested merged revisions. To explore the merged revisions for revision X, use the following command:

```
bzr log -n0 -rX
```

To see all revisions and all their merged revisions:

```
bzr log -n0
```

Note that the -n option is used to indicate the number of levels to display where 0 means all. If that is too noisy, you can easily adjust the number to only view down so far. For example, if your project is structured with a top level gatekeeper merging changes from team gatekeepers, bzr log shows what the top level gatekeeper did while bzr log -n2 shows what the team gatekeepers did. In the vast majority of cases though, -n0 is fine.

#### Tuning the output

The log command has several options that are useful for tuning the output. These include:

- --forward presents the log in chronological order, i.e. the most recent revisions are displayed last.
- the --limit option controls the maximum number of revisions displayed.

See the online help for the log command or the User Reference for more information on tuning the output.

### Viewing the history for a file

It is often useful to filter the history so that it only applies to a given file. To do this, provide the filename to the log command like this:

bzr log foo.py

#### Viewing an old version of a file

To get the contents of a file at a given version, use the cat command like this:

```
bzr cat -r X file
```

where x is the revision identifier and file is the filename. This will send output to the standard output stream so you'll typically want to pipe the output through a viewing tool (like less or more) or redirect it like this:

```
bzr cat -r -2 foo.py | less
bzr cat -r 1 foo.py > /tmp/foo-1st-version.py
```

#### Graphical history viewers

History browsing is one area where GUI tools really make life easier. Bazaar has numerous plug-ins that provide this capability including QBzr and bzr-gtk. See Using plugins for details on how to install these if they are not already installed.

To use the graphical viewer from QBzr:

bzr qlog

To use the graphical viewer from bzr-gtk:

bzr viz

viz is actually a built-in alias for visualize so use the longer command name if you prefer.

Home | Documentation | Table of Contents (2.2b1) » Bazaar User Guide » previous | next

# Releasing a project

#### Packaging a release

The export command is used to package a release, i.e. to take a copy of the files and directories in a branch and package them into a fresh directory or archive. For example, this command will package the last committed version into a tar.gz archive file:

```
bzr export ../releases/my-stuff-1.5.tar.gz
```

The export command uses the suffix of the archive file to work out the type of archive to create as shown below.

dir	(none)
tar	.tar
tbz2	.tar.bz2, .tbz2
tgz	.tar.gz, .tgz
tgz zip	.zip

If you wish to package a revision other than the last one, use the -r option. If you wish to tune the root directory inside the archive, use the --root option. See the online help or User Reference for further details on the options supported by export.

#### Tagging a release

Rather than remembering which version was used to package a release, it's useful to define a symbolic name for a version using the tag command like this:

```
bzr tag version-1-5
```

That tag can be used later whenever a revision identifier is required, e.g.:

```
bzr diff -r tag:version-1-5
```

To see the list of tags defined in a branch, use the tags command.

Home | Documentation | Table of Contents (2.2b1) » Bazaar User Guide » previous | next

## Undoing mistakes

## Mistakes happen

Bazaar has been designed to make it easy to recover from mistakes as explained below.

### Dropping the revision history for a project

If you accidentally put the wrong tree under version control, simply delete the <code>.bzr</code> directory.

#### Deregistering a file or directory

If you accidentally register a file using add that you don't want version controlled, you can use the remove command to tell Bazaar to forget about it.

remove has been designed to *Do the Safe Thing* in that it will not delete a modified file. For example:

```
bzr add foo.html
(oops - didn't mean that)
bzr remove foo.html
```

This will complain about the file being modified or unknown. If you want to keep the file, use the --keep option. Alternatively, if you want to delete the file, use the --force option. For example:

```
bzr add foo.html
(oops - didn't mean that)
bzr remove --keep foo.html
(foo.html left on disk, but deregistered)
```

On the other hand, the unchanged **TODO** file is deregistered and removed from disk without complaint in this example:

```
bzr add TODO
bzr commit -m "added TODO"
(hack, hack, hack - but don't change TODO)
bzr remove TODO
(TODO file deleted)
```

Note: If you delete a file using your file manager, IDE or via an operating system command, the commit command will implicitly treat it as removed.

#### Undoing changes since the last commit

One of the reasons for using a version control tool is that it lets you easily checkpoint good tree states while working. If you decide that the changes you have made since the last commit ought to be thrown away, the command to use is revert like this:

#### bzr revert

As a precaution, it is good practice to use bzr status and bzr diff first to check that everything being thrown away really ought to be.

# Undoing changes to a file since the last commit

If you want to undo changes to a particular file since the last commit but keep all the other changes in the tree, pass the filename as an argument to revert like this:

bzr revert foo.py

#### Undoing the last commit

If you make a commit and really didn't mean to, use the uncommit command to undo it like this:

```
bzr uncommit
```

Unlike revert, uncommit leaves the content of your working tree exactly as it is. That's really handy if you make a commit and accidently provide the wrong error message. For example:

```
bzr commit -m "Fix bug #11"
(damn - wrong bug number)
bzr uncommit
bzr commit -m "Fix bug #1"
```

Another common reason for undoing a commit is because you forgot to add one or more files. Some users like to alias commit to commit - strict so that commits fail if unknown files are found in the tree.

Note: While the merge command is not introduced until the next chapter, it is worth noting now that uncommit restores any pending merges. (Running bzr status after uncommit will show these.) merge can also be used to effectively undo just a selected commit earlier in history. For more information on merge, see Merging changes in the next chapter and the Bazaar User Reference.

#### Undoing multiple commits

You can use the -r option to undo several commits like this:

bzr uncommit -r -3

If your reason for doing this is that you really want to back out several changes, then be sure to remember that uncommit does not change your working tree: you'll probably need to run the revert command as well to complete the task. In many cases though, it's arguably better to leave your history alone and add a new revision reflecting the content of the last good state.

#### Reverting to the state of an earlier version

If you make an unwanted change but it doesn't make sense to uncommit it (because that code has been released to users say), you can use revert to take your working tree back to the desired state. For example:

```
% bzr commit "Fix bug #5"
Committed revision 20.
(release the code)
(hmm - bad fix)
bzr revert -r 19
bzr commit -m "Backout fix for bug #5"
```

This will change your entire tree back to the state as of revision 19, which is probably only what you want if you haven't made any new commits since then. If you have, the revert would wipe them out as well. In that case, you probably want to use Reverse cherrypicking instead to back out the bad fix.

Note: As an alternative to using an absolute revision number (like 19), you can specify one relative to the tip (-1) using a negative number like this:

```
bzr revert -r -2
```

## Correcting a tag

If you have defined a tag prematurely, use the --force option of the tag command to redefine it. For example:

```
bzr tag 2.0-beta-1
(oops, we're not yet ready for that)
(make more commits to include more fixes)
bzr tag 2.0-beta-1 --force
```

#### Clearing a tag

If you have defined a tag and no longer want it defined, use the --delete option of the tag command to remove it. For example:

```
bzr tag 2.0-beta-4
(oops, we're not releasing a 4th beta)
bzr tag 2.0-beta-4 --delete
```

Home | Documentation | Table of Contents (2.2b1) » Bazaar User Guide » previous | next

## Working with another

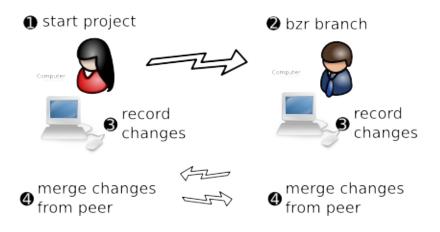
#### Peer-to-peer rocks

In many cases, two minds can be better than one. You may be the one who started a project and someone wishes to help, or perhaps it's you who wants to help another. Perhaps you are both members of a larger team that have been assigned a task together as pair programmers. Either way, two people need to agree on a process, a set of guidelines and a toolset in order to work together effectively.

Imagine if you were not allowed to call someone on the phone directly and the only way to talk to them was by registering a conference call first? Companies and communities that only share code via a central VCS repository are living with a similar straitjacket to that every day. There are times when central control makes a lot of sense and times when peer-to-peer rocks. Either way, Bazaar is designed to help.

#### The partner workflow

While it's certainly not the only way to do it, the *partner workflow* below is a good starting point for a pair of people who wish to collaborate using Bazaar.



Over and above the tasks covered in the previous chapter, this chapter introduces two essential collaboration activities:

- getting a copy of a branch
- merging changes between branches.

Even when it's just you working on a code base, it can be very useful to keep multiple branches around (for different releases say) and to merge changes between them as appropriate. Your "partner" may indeed be yourself.

## Branching a project

### **Branch URLs**

Before someone else can get a copy of your work, you need to agree on a transfer technology. You may decide to make the top level directory of your branch a network share, an approach familiar to Windows users. Linux and OS X users might prefer access to be via SFTP, a secure protocol built-in to most SSH servers. Bazaar is *very* flexible in this regard with support for lots of protocols some of which are given below.

Prefix	Description				
file://	Access using the standard filesystem (default)				
sftp://	Access using SFTP (most SSH servers provide SFTP).				
bzr://	Fast access using the Bazaar smart server.				
ftp://	Access using passive FTP.				
http://	Read-only access to branches exported by a web server.				

As indicated above, branches are identified using URLs with the prefix indicating the transfer technology. If no prefix is given, normal filenames are assumed. For a complete list of supported protocols, see the <u>urlspec</u> online help topic or the <u>URL Identifiers</u> section of the Bazaar User Reference.

URLs are normally resolved relative to the root directory of the server, so <a href="ftp://example.com/repo/foo">ftp://example.com/repo/foo</a> means the <a href="free-repo/foo">/repo/foo</a> directory of that host. (We say 'normally' because some server software like Apache can be configured to remap URLs arbitrarily, in which case you'll need to look at the server configuration to find out which URL corresponds to which directory.)

To address a path relative to your home directory on the server, use a tilde like so: sftp://example.com/~/public\_html should map to

public\_html within your home directory.

### A reminder about shared repositories

Before getting a copy of a branch, have a quick think about where to put it on your filesystem. For maximum storage efficiency down the track, it is recommended that branches be created somewhere under a directory that has been set up as a shared repository. (See Feature branches in Organizing your workspace for a commonly used layout.) For example:

```
bzr init-repo my-repo
cd my-repo
```

You are now ready to grab a branch from someone else and hack away.

### The branch command

To get a branch based on an existing branch, use the branch command. The syntax is:

```
bzr branch URL [directory]
```

If a directory is not given, one is created based on the last part of the URL. Here are some examples showing a drive qualified path (M:/) and an sftp URL respectively:

```
bzr branch M:/cool-trunk
bzr branch sftp://bill@mary-laptop/cool-repo/cool-trunk
```

This example shows explicitly giving the directory name to use for the new branch:

bzr branch /home/mary/cool-repo/cool-trunk cool

### Time and space considerations

Depending on the size of the branch being transferred and the speed and latency of the network between your computer and the source branch, this initial transfer might take some time. Subsequent updates should be much faster as only the changes are transferred then.

Keep in mind that Bazaar is transferring the complete history of the branch, not just the latest snapshot. As a consequence, you can be off the network (or disconnected from the network share) after branch completes but you'll still be able to log and diff the history of the branch as much as you want. Furthermore, these operations are quick as the history is stored locally.

Note that Bazaar uses smart compression technology to minimize the amount of disk space required to store version history. In many cases, the complete history of a project will take up less disk space than the working copy of the latest version.

As explained in later chapters, Bazaar also has support for lightweight checkouts of a branch, i.e. working trees with no local storage of history. Of course, disconnected usage is not available then but that's a tradeoff you can decide to make if local disk space is really tight for you. Support for limited lookback into history - history horizons - is currently under development as well.

## Viewing branch information

If you wish to see information about a branch including where it came from, use the info command. For example:

bzr info cool

If no branch is given, information on the current branch is displayed.

Home | Documentation | Table of Contents (2.2b1) » Bazaar User Guide » previous | next

# Merging changes

## Parallel development

Once someone has their own branch of a project, they can make and commit changes in parallel to any development proceeding on the original branch. Pretty soon though, these independent lines of development will need to be combined again. This process is known as *merging*.

### The merge command

To incorporate changes from another branch, use the merge command. Its syntax is:

```
bzr merge [URL]
```

If no URL is given, a default is used, initially the branch this branch originated from. For example, if Bill made a branch from Mary's work, he can merge her subsequent changes by simply typing this:

```
bzr merge
```

On the other hand, Mary might want to merge into her branch the work Bill has done in his. In this case, she needs to explicitly give the URL the first time, e.g.:

```
bzr merge sftp://mary@bill-laptop/cool-repo/cool-trunk
```

This sets the default merge branch if one is not already set. To change the default after it is set, use the --remember option.

## How does merging work?

A variety of algorithms exist for merging changes. Bazaar's default algorithm is a variation of *3-way merging* which works as follows. Given an ancestor A and two branches B and C, the following table provides the rules used.

Α	В	C	Result	Comment
X	Χ	Χ	Х	unchanged
X	Χ	У	У	line from C
X	у	Χ	У	line from B
Χ	У	Z	?	conflict

Note that some merges can only be completed with the assistance of a human. Details on how to resolve these are given in Resolving conflicts.

### Recording a merge

After any conflicts are resolved, the merge needs to be committed. For example:

```
bzr commit -m "Merged Mary's changes"
```

Even if there are no conflicts, an explicit commit is still required. Unlike some other tools, this is considered a feature in Bazaar. A clean merge is not necessarily a good merge so making the commit a separate explicit step allows you to run your test suite first to verify all is good. If problems are found, you should correct them before committing the merge or throw the merge away using revert.

## Merge tracking

One of the most important features of Bazaar is distributed, high quality *merge tracking*. In other words, Bazaar remembers what has been merged already and uses that information to intelligently choose the best ancestor for a merge, minimizing the number and size of conflicts.

If you are a refugee from many other VCS tools, it can be really hard to "unlearn" the *please-let-me-avoid-merging-at-any-cost* habit. Bazaar lets you safely merge as often as you like with other people. By working in a peer-to-peer manner when it makes sense to do so, you also avoid using a central branch as an "integration swamp", keeping its quality higher. When the change you're collaborating on is truly ready for wider sharing, that's the time to merge and commit it to a central branch, not before.

Merging that Just Works truly can change how developers work together.

# Resolving conflicts

### Workflow

Unlike some other tools that force you to resolve each conflict during the merge process, Bazaar merges as much as it can and then reports the conflicts. This can make conflict resolution easier because the contents of the whole post-merge tree are available to help you decide how things ought to be resolved. You may also wish to selectively run tests as you go to confirm each resolution or group or resolutions is good.

## Listing conflicts

As well as being reported by the merge command, the list of outstanding conflicts may be displayed at any time by using the conflicts command. It is also included as part of the output from the status command.

### Resolving a conflict

When a conflict is encountered, the merge command puts embedded markers in each file showing the areas it couldn't resolve. It also creates 3 files for each file with a conflict:

- foo.BASE
- foo.THIS
- foo.OTHER

where foo is the name of the conflicted file. In many cases, you can resolve conflicts by simply manually editing each file in question, fixing the relevant areas and removing the conflict markers as you go.

After fixing all the files in conflict, and removing the markers, ask Bazaar to mark them as resolved using the resolve command like this:

#### bzr resolve

Alternatively, after fixing each file, you can mark it as resolved like this:

#### bzr resolve foo

Among other things, the resolve command cleans up the BASE, THIS and OTHER files from your working tree.

### Using the remerge command

In some cases, you may wish to try a different merge algorithm on a given file. To do this, use the remerge command nominating the file like this:

bzr remerge --weave foo

where foo is the file and weave is one of the available merge algorithms. This algorithm is particularly useful when a so-called criss-cross merge is detected, e.g. when two branches merge the same thing then merge each other. See the online help for criss-cross and remerge for further details.

### Using external tools to resolve conflicts

If you have a GUI tool you like using to resolve conflicts, be sure to install the *extmerge* plugin. Once installed, it can be used like this:

bzr extmerge foo

where foo is the conflicted file. Rather than provide a list of files to resolve, you can give the --all option to implicitly specify all conflicted files.

The extmerge command uses the tool specified by the external\_merge setting in your bazaar.conf file. If not set, it will look for some popular merge tools such as kdiff3 or opendiff, the latter being a command line interface to the FileMerge utility in OS X.

Home | Documentation | Table of Contents (2.2b1) » Bazaar User Guide » previous | next

## Annotating changes

### Seeing the origin of content

When two or more people are working on files, it can be highly useful at times to see who created or last changed certain content. To do this, using the annotate command like this:

bzr annotate readme.txt

If you are a pessimist or an optimist, you might prefer to use one of built-in aliases for annotate: blame or praise. Either way, this displays each line of the file together with information such as:

- who changed it last
- when it was last changed
- the commit message.

### **GUI** tools

The various GUI plugins for Bazaar provide graphical tools for viewing annotation information. For example, the bzr-gtk plugin provides a GUI tool for this that can be launched using the gannotate command:

bzr gannotate readme.txt

The GUI tools typically provide a much richer display of interesting information (e.g. all the changes in each commit) so you may prefer them over the text-based command.

Home | Documentation | Table of Contents (2.2b1) » Bazaar User Guide » previous | next

# Centralized development

### Motivation

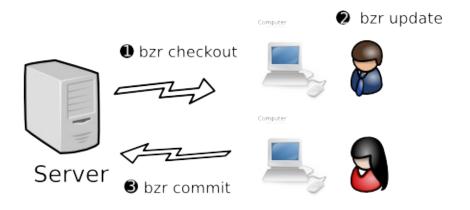
Rather than working in parallel and occasionally merging, it can be useful at times to work in lockstep, i.e. for multiple people to be continuously committing changes to a central location, merging their work with the latest content before every commit.

This workflow is very familiar to users of central VCS tools like Subversion and CVS. It is also applicable to a single developer who works on multiple machines, e.g. someone who normally works on a desktop computer but travels with a laptop, or someone who uses their (Internet connected) home computer to complete office work out of hours.

If centralized development works well for your team already, that's great. Many teams begin using Bazaar this way and experiment with alternative workflows later.

### Centralized workflow

The diagram below provides an overview of the *centralized workflow*.



Even if your team is planning to use a more distributed workflow, many of the tasks covered in this chapter may be useful to you, particularly how to publish branches.

# Publishing a branch

### Setting up a central repository

While the centralized workflow can be used by socially nominating any branch on any computer as the central one, in practice most teams have a dedicated server for hosting central branches.

Just as it's best practice to use a shared repository locally, it's advisable to put central branches in a shared repository. Note that central shared branches typically only want to store history, not working copies of files, so their enclosing repository is usually creating using the no-trees option, e.g.:

bzr init-repo --no-trees sftp://centralhost/srv/bzr/PROJECT

You can think of this step as similar to setting up a new cvsroot or Subversion repository. However, Bazaar gives you more flexibility in how branches may be organised in your repository. See Advanced shared repository layouts in the appendices for guidelines and examples.

### Starting a central branch

There are two ways of populating a central branch with some initial content:

- 1. Making a local branch and pushing it to a central location
- 2. Making an empty central branch then committing content to it.

Here is an example of the first way:

Here is an example of the second way:

Note that committing inside a working tree created using the checkout command implicitly commits the content to the central location as well as locally. Had we used the branch command instead of checkout above, the content would have only been

committed locally.

Working trees that are tightly bound to a central location like this are called *checkouts*. The rest of this chapter explains their numerous features in more detail.

Home | Documentation | Table of Contents (2.2b1) » Bazaar User Guide » previous | next

# Using checkouts

## Turning a branch into a checkout

If you have a local branch and wish to make it a checkout, use the bind command like this:

bzr bind sftp://centralhost/srv/bzr/PROJECT/trunk

This is necessary, for example, after creating a central branch using push as illustrated in the previous section.

After this, commits will be applied to the bound branch before being applied locally.

### Turning a checkout into a branch

If you have a checkout and wish to make it a normal branch, use the unbind command like this:

bzr unbind

After this, commits will only be applied locally.

#### Getting a checkout

When working in a team using a central branch, one person needs to provide some initial content as shown in the previous section. After that, each person should use the <a href="https://checkout.com/checkout">checkout</a> command to create their local checkout, i.e. the sandbox in which they will make their changes.

Unlike Subversion and CVS, in Bazaar the checkout command creates a local full copy of history in addition to creating a working tree holding the latest content. This means that operations such as diff and log are fast and can still be used when disconnected from the central location.

#### Getting a lightweight checkout

While Bazaar does its best to efficiently store version history, there are occasions when the history is simply not wanted. For example, if your team is managing the content of a web site using Bazaar with a central repository, then your release process might be as simple as updating a checkout of the content on the public web server. In this case, you probably don't want the history downloaded to that location as doing so:

- wastes disk space holding history that isn't needed there
- exposes a Bazaar branch that you may want kept private.

To get a history-less checkout in Bazaar, use the --lightweight option like this:

```
bzr checkout --lightweight sftp://centralhost/srv/bzr/PROJECT/t
```

Of course, many of the benefits of a normal checkout are lost by doing this but that's a tradeoff you can make if and when it makes sense.

The --lightweight option only applies to checkouts, not to all branches.

Note: If your code base is really large and disk space on your computer is limited, lightweight checkouts may be the right choice for you. Be sure to consider all your options though including shared repositories, stacked branches, and reusing a checkout.

### Updating to the latest content

One of the important aspects of working in lockstep with others is keeping your checkout up to date with the latest changes made to the central branch. Just as you would in Subversion or CVS, you do this in Bazaar by using the update command like this:

#### bzr update

This gets any new revisions available in the bound branch and merges your local changes, if any.

#### Handling commit failures

Note that your checkout *must* be up to date with the bound branch before running commit. Bazaar is actually stricter about this than Subversion or CVS - you need to be up to date with the full tree, not just for the files you've changed. Bazaar will ask you to run update if it detects that a revision has been added to the central location since you last updated.

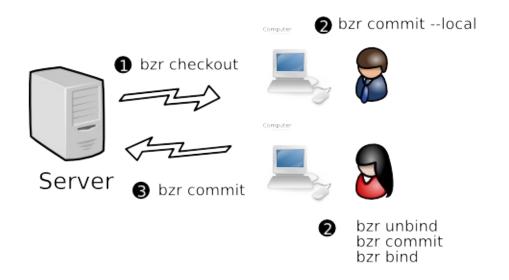
If the network connection to the bound branch is lost, the commit will fail. Some alternative ways of working around that are outlined next.

Home | Documentation | Table of Contents (2.2b1) » Bazaar User Guide » previous | next

## Working offline on a central branch

#### The centralized with local commits workflow

If you lose your network connection because you are travelling, the central server goes down, or you simply want to snapshot changes locally without publishing them centrally just yet, this workflow is for you.



### **Committing locally**

If you're working in a checkout and need/wish to commit locally only, add the --local option to the commit command like this:

bzr commit --local

#### Being disconnected for long time periods

If you will be or want to be disconnected from the bound branch for a while, then remembering to add --local to every commit command can be annoying. An alternative is to use the unbind command to make the checkout temporarily into a normal branch followed by the bind command at some later point in time when you want to keep in lockstep again.

Note that the bind command remembers where you were bound to last time this branch was a checkout so it isn't necessary to enter the URL of the remote branch when you use bind after an earlier unbind.

### Merging a series of local commits

When you make commits locally independent of ongoing development on a central branch, then Bazaar treats these as two lines of development next time you update. In this case, update does the following:

- it brings the latest revisions from the bound branch down and makes that the mainline of development within your checkout
- it moves your local changes since you last updated into a logical parallel branch
- it merges these together so that your local changes are reported as a pending merge by status.

As always, you will need to run commit after this to send your work to the central branch.

Home | Documentation | Table of Contents (2.2b1) » Bazaar User Guide » previous | next

## Reusing a checkout

#### Motivation

At times, it can be useful to have a single checkout as your sandbox for working on multiple branches. Some possible reasons for this include:

- saving disk space when the working tree is large
- developing in a fixed location.

In many cases, working tree disk usage swamps the size of the .bzr directory. If you want to work on multiple branches but can't afford the overhead of a full working tree for each, reusing a checkout across multiples branches is the way to go.

On other occasions, the location of your sandbox might be configured into numerous development and testing tools. Once again, reusing a checkout across multiple branches can help.

#### Changing where a branch is bound to

To change where a checkout is bound to, follow these steps:

- 1. Make sure that any local changes have been committed centrally so that no work is lost.
- 2. Use the bind command giving the URL of the new remote branch you wish to work on.
- 3. Make your checkout a copy of the desired branch by using the update command followed by the revert command.

Note that simply binding to a new branch and running update merges in your local changes, both committed and uncommitted. You need to decide whether to keep them or not by running either revert or commit.

An alternative to the bind+update recipe is using the switch command. This is basically the same as removing the existing branch and running checkout again on the new location, except that any uncommitted changes in your tree are merged in.

Note: As switch can potentially throw away committed changes in order to make a checkout an accurate cache of a different bound branch, it will fail by design if there are changes which have been committed locally but are not yet committed to the most recently bound branch. To truly abandon these changes, use the --force option.

#### Switching a lightweight checkout

With a lightweight checkout, there are no local commits and switch effectively changes which branch the working tree is associated with. One possible setup is to use a lightweight checkout in combination with a local tree-less repository. This lets you switch what you are working on with ease. For example:

```
bzr init-repo --no-trees PROJECT
cd PROJECT
bzr branch sftp://centralhost/srv/bzr/PROJECT/trunk
bzr checkout --lightweight trunk my-sandbox
cd my-sandbox
(hack away)
```

Note that trunk in this example will have a .bzr directory within it but there will be no working tree there as the branch was created in a tree-less repository. You can grab or create as many branches as you need there and switch between them as required. For example:

```
(assuming in my-sandbox)
bzr branch sftp://centralhost/srv/bzr/PROJECT/PROJECT-1.0 ../PR
bzr switch ../PROJECT-1.0
(fix bug in 1.0)
bzr commit -m "blah, blah blah"
bzr switch ../trunk
(go back to working on the trunk)

▶
```

Note: The branches may be local only or they may be bound to remote ones (by creating them with checkout or by using bind after creating them with branch).

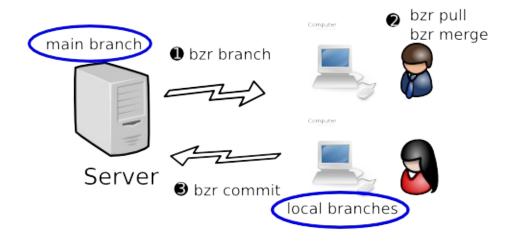
## Distributed development

#### Motivation

Distributed VCS tools offer new ways of working together, ways that better reflect the modern world we live in and ways that enable higher quality outcomes.

# The decentralized with shared mainline workflow

In this workflow, each developer has their own branch or branches, plus a checkout of the main branch. They do their work in their personal branch, then merge it into the mainline when it is ready.



Other distributed workflows are explored later in this chapter.

# Organizing branches

#### Mirror branches

A primary difference when using distributed workflows to develop is that your main local branch is not the place to make changes. Instead, it is kept as a pristine copy of the central branch, i.e. it's a *mirror branch*.

To create a mirror branch, set-up a shared repository (if you haven't already) and then use the branch (or checkout) command to create the mirror. For example:

bzr init-repo PROJECT
cd PROJECT
bzr branch sftp://centralhost/srv/bzr/PROJECT/trunk

#### Task branches

Each new feature or fix is developed in its own branch. These branches are referred to as *feature branches* or *task branches* - the terms are used interchangeably.

To create a task branch, use the branch command against your mirror branch. For example:

```
bzr branch trunk fix-123
cd fix-123
(hack, hack, hack)
```

There are numerous advantages to this approach:

- 1. You can work on multiple changes in parallel
- 2. There is reduced coupling between changes
- 3. Multiple people can work in a peer-to-peer mode on a branch until it is ready to go.

In particular, some changes take longer to cook than others so you can ask for reviews, apply feedback, ask for another review, etc. By completing work to sufficient quality in separate branches before merging into a central branch, the quality and stability of the central branch are maintained at higher level than they otherwise would be.

### Refreshing a mirror branch

Use the pull command to do this:

cd trunk bzr pull

# Merging the latest trunk into a feature branch

Use the merge command to do this:

```
cd fix-123
bzr merge
(resolve any conflicts)
bzr commit -m "merged trunk"
```

#### Merging a feature into the trunk

The policies for different distributed workflows vary here. The simple case where all developers have commit rights to the main trunk are shown below.

If your mirror is a checkout:

```
cd trunk
bzr update
bzr merge ../fix-123
(resolve any conflicts)
bzr commit -m "Fixed bug #123"
```

If your mirror is a branch:

```
cd trunk
bzr pull
bzr merge ../fix-123
(resolve any conflicts)
bzr commit -m "Fixed bug #123"
bzr push
```

#### Backing up task branches

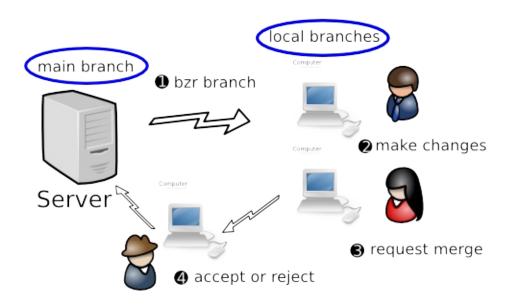
One of the side effects of centralized workflows is that changes get frequently committed to a central location which is backed up as part of normal IT operations. When developing on task branches, it is a good idea to publish your work to a central location (but not necessarily a shared location) that will be backed up. You may even wish to bind local task branches to remote ones established on a backup server just for this purpose.

Home | Documentation | Table of Contents (2.2b1) » Bazaar User Guide » previous | next

## Using gatekeepers

# The decentralized with human gatekeeper workflow

In this workflow, one developer (the gatekeeper) has commit rights to the main branch while other developers have read-only access. All developers make their changes in task branches.



When a developer wants their work merged, they ask the gatekeeper to review their change and merge it if acceptable. If a change fails review, further development proceeds in the relevant task branch until it is good to go.

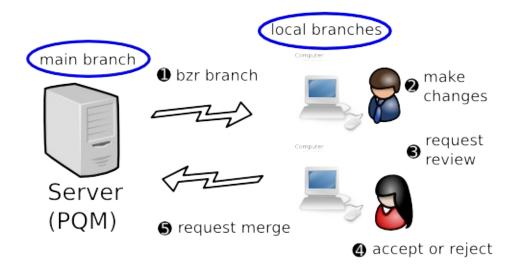
Note that a key aspect of this approach is the inversion of control that is implied: developers no longer decide when to "commit/push" changes into the central branch: the code base evolves by gatekeepers "merging/pulling" changes in a controlled manner. It's perfectly acceptable, indeed common, to have multiple central branches with different gatekeepers, e.g. one branch for the current

production release and another for the next release. In this case, a task branch holding a bug fix will most likely be advertised to both gatekeepers.

One of the great things about this workflow is that it is hugely scalable. Large projects can be broken into teams and each team can have a *local master branch* managed by a gatekeeper. Someone can be appointed as the primary gatekeeper to merge changes from the team master branches into the primary master branch when team leaders request it.

# The decentralized with automatic gatekeeper workflow

To obtain even higher quality, all developers can be required to submit changes to an automated gatekeeper that only merges and commits a change if it passes a regression test suite. One such gatekeeper is a software tool called PQM.



For further information on PQM, see https://launchpad.net/pqm.

## Sending changes

#### Motivation

In many distributed development scenarios, it isn't always feasible for developers to share task branches by advertising their URLs. For example, a developer working on a laptop might take it home overnight so his/her task branches could well be inaccessible when a gatekeeper in another timezone wants to review or merge it.

Bazaar provides a neat feature to assist here: *merge directives*.

#### Understanding merge directives

You can think of a merge directive as a "mini branch" - just the new growth on a branch since it was created. It's a software patch showing what's new but with added intelligence: metadata like interim commits, renames and digital signatures.

Another useful metaphor is a packet cake: a merge directive has a recipe together with the ingredients you need bundled inside it. To stretch the metaphor, the ingredients are all the metadata on the changes made to the branch; the recipe is instructions on how those changes ought to be merged, i.e. information for the merge command to use in selecting common ancestors.

Regardless of how you think of them, merge directives are neat. They are easy to create, suitable for mailing around as attachments and can be processed much like branches can on the receiving end.

# Creating a merge directive

To create and optionally send a merge directive, use the send command.

By default, send will email the merge directive to the "submission address" for the branch, which is typically the lead developer or the development mailing list. send without options will create a merge directive, fire up your email tool and attach it, ready for you to add the explanatory text bit. (See the online help for send and Configuration Settings in the User Reference for further details on how to configure this.)

Most projects like people to add some explanation to the mail along with the patch, explaining the reason for the patch, and why it is done the way it is. This gives a reviewer some context before going into the line-by-line diff.

Alternatively, if the --output (or -o) option is given, send will write the merge directive to a file, so you can mail it yourself, examine it, or save it for later use. If an output file of - is given, the directive is written to stdout. For example:

```
cd X-fix-123
bzr send -o ../fix-123.patch
```

# Applying a merge directive

Merge directives can be applied in much the same way as branches: by using the merge and pull commands.

They can also be useful when communicating with upstream projects that don't use Bazaar. In particular, the preview of the overall change in a merge directive looks like a vanilla software patch, so they can be applied using patch -po for example.

Home | Documentation | Table of Contents (2.2b1) » Bazaar User Guide » previous | next

# The journey ahead

We hope that earlier chapters have given you a solid understanding of how Bazaar can assist you in being productive on your own and working effectively with others. If you are learning Bazaar for the first time, it might be good to try the procedures covered already for a while, coming back to this manual once you have mastered them.

Remaining chapters covers various topics to guide you in further optimizing how you use Bazaar. Unless stated otherwise, the topics in this and remaining chapters are independent of each other and can therefore be read in whichever order you wish.

Home | Documentation | Table of Contents (2.2b1) » Bazaar User Guide » previous | next

# Pseudo merging

#### Cherrypicking

At times, it can be useful to selectively merge some of the changes in a branch, but not all of them. This is commonly referred to as *cherrypicking*. Here are some examples of where cherrypicking is useful:

- selectively taking fixes from the main development branch into a release branch
- selectively taking improvements out of an experimental branch into a feature branch.

To merge only the changes made by revision X in branch foo, the command is:

```
bzr merge -c X foo
```

To merge only the changes up to revision X in branch foo, the command is:

```
bzr merge -r X foo
```

To merge only the changes since revision X in branch foo, the command is:

```
bzr merge -r X.. foo
```

To merge only the changes from revision X to revision Y in branch foo, the command is:

```
bzr merge -r X..Y foo
```

Like a normal merge, you must explicitly commit a cherrypick. You may wish to see the changes made using bzr diff, and run your

test suite if any, before doing this.

Unlike a normal merge, Bazaar does not currently track cherrypicks. In particular, the changes look like a normal commit and the (internal) revision history of the changes from the other branch is lost. In many cases where they are useful (see above), this is not a major problem because there are good reasons why a full merge should never be done at a later time. In other cases, additional conflicts will need to be resolved when the changes are merged again.

## Merging without parents

A related technique to cherrypicking, in that it makes changes without reference to the revisions that they came from is to perform a merge, but forget about the parent revisions before committing. This has the effect of making all of the changes that would have been in the merge happen in a single commit. After the merge and before the corresponding commit, you can do:

#### bzr revert --forget-merges

to keep the changes in the working tree, but remove the record of the revisions where the changes originated. The next commit would then record all of those changes without any record of the merged revisions.

This is desired by some users to make their history "cleaner", but you should be careful that the loss of history does not outweigh the value of cleanliness, particularly given Bazaar's capabilities for progressively disclosing merged revisions. In particular, because this will include the changes from the source branch, but without attribution to that branch, it can lead to additional conflicts on later merges that involve the same source and target branches.

## Reverse cherrypicking

Cherrypicking can be used to reverse a set of changes made by giving an upper bound in the revision range which is *below* the lower bound. For example, to back-out changes made in revision 10, the command is:

```
bzr merge -r 10..9
```

If you want to take most changes, but not all, from somewhere else, you may wish to do a normal merge followed by a few reverse cherrypicks.

# Merging uncommitted changes

If you have several branches and you accidentally start making changes in the wrong one, here are the steps to take to correct this. Assuming you began working in branch foo when you meant to work in branch bar:

- 1. Change into branch bar.
- 2. Run bzr merge --uncommitted foo
- 3. Check the changes came across (bzr diff)
- 4. Change into branch foo
- 5. Run bzr revert.

#### Rebasing

Another option to normal merging is *rebasing*, i.e. making it look like the current branch originated from a different point than it did. Rebasing is supported in Bazaar by the rebase command provided by the rebase plugin.

The rebase command takes the location of another branch on which the branch in the current working directory will be rebased. If a branch is not specified then the parent branch is used, and this is usually the desired result.

The first step identifies the revisions that are in the current branch that are not in the parent branch. The current branch is then set to be at the same revision as the target branch, and each revision is replayed on top of the branch. At the end of the process it will appear as though your current branch was branched off the current last revision of the target.

Each revision that is replayed may cause conflicts in the tree. If this happens the command will stop and allow you to fix them up. Resolve the commits as you would for a merge, and then run bzr resolve to marked them as resolved. Once you have resolved all the conflicts, you should run bzr rebase-continue to continue the rebase operation. If conflicts are encountered and you decide not to continue, you can run bzr rebase-abort. You can also use rebase-todo to show the list of commits still to be replayed.

Note: Some users coming from central VCS tools with poor merge tracking like rebasing because it's similar to how they are use to working in older tools, or because "perfectly clean" history seems important. Before rebasing in Bazaar, think about whether a normal merge is a better choice. In particular, rebasing a private branch

before sharing it is OK but rebasing after sharing a branch with someone else is **strongly** discouraged.

Home | Documentation | Table of Contents (2.2b1) » Bazaar User Guide » previous | next

# **Shelving Changes**

Sometimes you will want to temporarily remove changes from your working tree and restore them later, For instance to commit a small bug-fix you found while working on something. Bazaar allows you to put changes on a shelf to achieve this. When you want to restore the changes later you can use unshelve to apply them to your working tree again.

For example, consider a working tree with one or more changes made ...

```
$ bzr diff
=== modified file 'description.txt'
--- description.txt
+++ description.txt
@@ -2,7 +2,7 @@
 ==========
These plugins
-by Michael Ellerman
+written by Michael Ellerman
 provide a very
fine-grained 'undo'
 facility
@@ -11,6 +11,6 @@
 This allows you to
 undo some of
 your changes,
-commit, and get
+perform a commit, and get
 back to where you
 were before.
```

The shelve command interactively asks which changes you want to retain in the working tree:

```
$ bzr shelve
--- description.txt
+++ description.txt
```

```
@@ -2,7 +2,7 @@
 ===========
These plugins
-by Michael Ellerman
+written by Michael Ellerman
 provide a very
 fine-grained 'undo'
 facility
Shelve? [yNfrq?]: y
--- description.txt
+++ description.txt
@@ -11,6 +11,6 @@
 This allows you to
 undo some of
 your changes,
-commit, and get
+perform a commit, and get
 back to where you
 were before.
Shelve? [yNfrq?]: n
Shelve 2 change(s)? [yNfrq?]', 'y'
Selected changes:
 M description.txt
Changes shelved with id "1".
```

If there are lots of changes in the working tree, you can provide the shelve command with a list of files and you will only be asked about changes in those files. After shelving changes, it's a good idea to use diff to confirm the tree has just the changes you expect:

```
$ bzr diff
=== modified file 'description.txt'
--- description.txt
+++ description.txt
@@ -2,7 +2,7 @@
===========

These plugins
-by Michael Ellerman
+written by Michael Ellerman
provide a very
```

```
fine-grained 'undo'
facility
```

Great - you're ready to commit:

```
$ bzr commit -m "improve first sentence"
```

At some later time, you can bring the shelved changes back into the working tree using unshelve:

```
$ bzr unshelve
Unshelving changes with id "1".
   M description.txt
All changes applied successfully.
```

If you want to, you can put multiple items on the shelf. Normally each time you run unshelve the most recently shelved changes will be reinstated. However, you can also unshelve changes in a different order by explicitly specifying which changes to unshelve.

Bazaar merges the changes in to your working tree, so they will apply even if you have edited the files since you shelved them, though they may conflict, in which case you will have to resolve the conflicts in the same way you do after a conflicted merge.

# Filtered views

#### Introducing filtered views

Views provide a mask over the tree so that users can focus on a subset of a tree when doing their work. There are several cases where this masking can be helpful. For example, technical writers and testers on many large projects may prefer to deal with just the directories/files in the project of interest to them.

Developers may also wish to break a large set of changes into multiple commits by using views. While shelve and unshelve let developers put some changes aside for a later commit, views let developers specify what to include in (instead of exclude from) the next commit.

After creating a view, commands that support a list of files - status, diff, commit, etc - effectively have that list of files implicitly given each time. An explicit list of files can still be given to these commands but the nominated files must be within the current view. In contrast, tree-centric commands - pull, merge, update, etc. - continue to operate on the whole tree but only report changes relevant to the current view. In both cases, Bazaar notifies the user each time it uses a view implicitly so that it is clear that the operation or output is being masked accordingly.

Note: Filtered views are only supported in format 2a, the default in Bazaar 2.0, or later.

# Creating a view

This is done by specifying the files and directories using the view command like this:

```
bzr view file1 file2 dir1 ...
```

The output is:

```
Using 'my' view: file1, file2, dir1
```

# Listing the current view

To see the current view, use the view command without arguments:

```
bzr view
```

If no view is current, a message will be output saying No current view. Otherwise the name and content of the current view will be displayed like this:

```
'my' view is: a, b, c
```

## Switching between views

In most cases, a view has a short life-span: it is created to make a selected change and is deleted once that change is committed. At other times, you may wish to create one or more named views and switch between them.

To define a named view and switch to it:

```
bzr view --name view-name file1 dir1 ...
```

#### For example:

```
bzr view --name doc NEWS doc/
Using doc view: NEWS, doc/
```

To list a named view:

```
bzr view --name view-name
```

To switch to a named view:

```
bzr view --switch view-name
```

To list all views defined:

```
bzr view --all
```

#### Temporarily disabling a view

To disable the current view without deleting it, you can switch to the pseudo view called off. This can be useful when you need to see the whole tree for an operation or two (e.g. merge) but want to switch back to your view after that.

To disable the current view without deleting it:

```
bzr view --switch off
```

After doing the operations you need to, you can switch back to the view you were using by name. For example, if the previous view used the default name:

bzr view --switch my

# **Deleting views**

To delete the current view:

```
bzr view --delete
```

To delete a named view:

```
bzr view --name view-name --delete
```

To delete all views:

```
bzr view --delete --all
```

### Things to be aware of

Defining a view does not delete the other files in the working tree - it merely provides a "lens" over the working tree.

Views are stored as working tree metadata. They are not propagated by branch commands like pull, push and update.

Views are defined in terms of file paths. If you move a file in a view to a location outside of the view, the view will no longer track that path. For example, if a view is defined as doc/ and doc/NEWS gets moved to NEWS, the views stays defined as doc/ and does not get changed to doc/ NEWS. Likewise, deleting a file in a view does not remove the file from that view.

The commands that use the current view are:

- status
- diff
- commit
- add
- remove
- revert
- mv
- Is.

Commands that operate on the full tree but only report changes inside the current view are:

- pull
- update
- merge.

Many commands currently ignore the current view. Over time, some

of these commands may be added to the lists above as the need arises. By design, some commands will most likely always ignore the current view because showing the whole picture is the better thing to do. Commands in this group include:

- log
- info.

Home | Documentation | Table of Contents (2.2b1) » Bazaar User Guide » previous | next

# Using stacked branches

#### Motivation

If you are working on a project, and you have read access to whose public repository but do not have write access to it, using stacked branches to backup/publish your work onto the same host of the public repository might be an option for you.

Other scenarios for stacked branch usage include experimental branches and code hosting sites. For these scenarios, stacked branches are ideal because of the benefits it provides.

#### What is a stacked branch?

A stacked branch is a branch that knows how to find revisions in another branch (the stacked-on branch). Stacked branches store just the unique revisions that are not in the stacked-on branch, making them faster to create and more storage efficient. In these respects, stacked branches are similar to shared repositories. However, stacked branches have additional benefits:

- The new branch can be in a completely different location to the branch being stacked on.
- Deleting the stacked branch really deletes the revisions (rather than leaving them in a shared repository).
- Security is improved over shared repositories, because the stacked-on repository can be physically readonly to developers committing to stacked branches.

## Creating a stacked branch

To create a stacked branch, use the stacked option of the branch command. For example:

bzr branch -- stacked source-url my-dir

This will create my-dir as a stacked branch with no local revisions. If it is defined, the public branch associated with source-url will be used as the *stacked-on* location. Otherwise, source-url will be the *stacked-on* location.

# Creating a stacked checkout

Direct creation of a stacked checkout is expected to be supported soon. In the meantime, a two step process is required:

- 1. Create a stacked branch as shown above.
- 2. Convert the branch into a checkout using either the reconfigure or bind command.

#### Pushing a stacked branch

Most changes on most projects build on an existing branch such as the *development trunk* or *current stable* branch. Creating a new branch stacked on one of these is easy to do using the push command like this:

```
bzr push --stacked-on reference-url my-url
```

This creates a new branch at my-url that is stacked on reference-url and only contains the revisions in the current branch that are not already in the branch at reference-url. In particular, my-url and reference-url can be on the same host, and the --stacked option can be used additionally to inform push to reference the revisions in reference-url. For example:

```
bzr push --stacked-on sftp://host/project --stacked sftp://host
```

This usage fits the scenario described in the Motivation section.

#### Limitations of stacked branches

Currently, you cannot commit to a stacked branch, due to bug 375013.

The important thing to remember about a stacked branch is that the stacked-on branch needs to be available for almost all operations. This is not an issue when both branches are local or both branches are on the same server.

Similarly, because most of the history is stored in the stacked-on repository, operations like bzr log can be slower when the stacked-on repository is accessed via a network.

## Changing branch stacking

Stacking of existing branches can be changed using the bzr reconfigure command to either stack on an existing branch, or to turn off stacking. Be aware that when bzr reconfigure --unstacked is used, bzr will copy all the referenced data from the stacked-on repository into the previously stacked repository. For large repositories this may take considerable time and may substantially increase the size of the repository.

# Running a smart server

Bazaar does not require a specialised server because it operates over HTTP, FTP or SFTP. There is an optional smart server that can be invoked over SSH, from inetd, or in a dedicated mode.

#### **Dumb servers**

We describe HTTP, FTP, SFTP and HTTP-WebDAV as "dumb" servers because they do not offer any assistance to Bazaar. If you make a Bazaar repository available over any of these protocols, Bazaar will allow you to read it remotely. Just enter the URL to the branch in the Bazaar command you are running.:

bzr log http://bazaar-vcs.org/bzr/bzr.dev

Bazaar supports writing over FTP, SFTP and (via a plugin) over HTTP-WebDAV.

### High-performance smart server

The high-performance smart server (hpss) performs certain operations much faster than dumb servers are capable of. In future releases, the range of operations that are improved by using the smart server will increase as we continue to tune performance.

To maintain the highest security possible, the current smart server provides read-only access by default. To enable read-write access, run it with --allow-writes. When using the SSH access method, bzr automatically runs with the --allow-writes option.

The alternative ways of configuring a smart server are explained below.

#### SSH

Using Bazaar over SSH requires no special configuration on the server; so long as Bazaar is installed on the server you can use bzr+ssh URLs, e.g.:

```
bzr log bzr+ssh://host/path/to/branch
```

If *bzr* is not installed system-wide on the server you may need to explicitly tell the local *bzr* where to find the remote *bzr*:

```
BZR_REMOTE_PATH=~/bin/bzr bzr log bzr+ssh://host/path/to/branch
```

The BZR\_REMOTE\_PATH environment variable adjusts how *bzr* will be invoked on the remote system. By default, just *bzr* will be invoked, which requires the *bzr* executable to be on the default search path. You can also set this permanently per-location in locations.conf.

Like SFTP, paths starting with ~ are relative to your home directory, e.g. bzr+ssh://example.com/~/code/proj. Additionally, paths starting with ~user will be relative to that user's home directory.

#### inetd

This example shows how to run *bzr* with a dedicated user *bzruser* for a shared repository in /srv/bzr/repo which has a branch at /srv/bzr/repo/branchname.

Running a Bazaar server from inetd requires an inetd.conf entry:

```
4155 stream tcp nowait bzruser /usr/bin/bzr /usr/bin/bzr s
```

When running client commands, the URL you supply is a *bzr://* URL relative to the --directory option given in inetd.conf:

```
bzr log bzr://host/branchname
```

If possible, paths starting with ~ and ~user will be expanded as for bzr+ssh. Home directories outside the --directory specified to bzr serve will not be accessible.

#### **Dedicated**

This mode has the same path and URL behaviour as the inetd mode. To run as a specific user, you should use su or login as that user.

This example runs bzr on its official port number of *4155* and listens on all interfaces. This allows connections from anywhere in the world that can reach your machine on port *4155*.

#### server:

bzr serve --directory=/srv/bzr/repo

#### client:

bzr log bzr://host/branchname

This example runs bzr serve on localhost port 1234.

#### server:

bzr serve --port=localhost:1234 --directory=/srv/bzr/repo

#### client:

bzr log bzr://localhost:1234/branchname

# Using hooks

#### What is a hook?

One way to customize Bazaar's behaviour is with *hooks*. Hooks allow you to perform actions before or after certain Bazaar operations. The operations include commit, push, pull, and uncommit. For a complete list of hooks and their parameters, see Hooks in the User Reference.

Most hooks are run on the client, but a few are run on the server. (Also see the push-and-update plugin that handles one special case of server-side operations.)

### Using hooks

To use a hook, you should write a plugin. Instead of creating a new command, this plugin will define and install the hook. Here's an example:

To use this example, create a file named <code>push\_hook.py</code>, and stick it in <code>plugins</code> subdirectory of your configuration directory. (If you have never installed any plugins, you may need to create the <code>plugins</code> directory).

That's it! The next time you push, it should show "The new revno is...". Of course, hooks can be much more elaborate than this, because you have the full power of Python at your disposal. Now that you know how to use hooks, what you do with them is up to you.

The plugin code does two things. First, it defines a function that will be run after push completes. (It could instead use an instance method or a callable object.) All push hooks take a single argument, the push\_result.

Second, the plugin installs the hook. The first argument 'post\_push' identifies where to install the hook. The second argument is the hook itself. The third argument is a name 'My post\_push hook', which can be used in progress messages and error messages.

## **Debugging hooks**

To get a list of installed hooks, use the hidden hooks command:

bzr hooks

# **Exporting version information**

## Getting the last revision number

If you only need the last revision number in your build scripts, you can use the revno command to get that value like this:

\$ bzr revno
3104

### Getting more version information

The version-info command can be used to output more information about the latest version like this:

```
$ bzr version-info
revision-id: pqm@pqm.ubuntu.com-20071211175118-s94sizduj201hrs5
date: 2007-12-11 17:51:18 +0000
build-date: 2007-12-13 13:14:51 +1000
revno: 3104
branch-nick: bzr.dev
■
```

You can easily filter that output using operating system tools or scripts. For example (on Linux/Unix):

```
$ bzr version-info | grep ^date
date: 2007-12-11 17:51:18 +0000
```

The --all option will actually dump version information about every revision if you need that information for more advanced post-processing.

### Python projects

If using a Makefile to build your project, you can generate the version information file as simply as:

```
library/_version.py:
    bzr version-info --format python > library/_version.py
```

This generates a file which contains 3 dictionaries:

- *version\_info*: A dictionary containing the basic information about the current state.
- revisions: A dictionary listing all of the revisions in the history of the tree, along with the commit times and commit message. This defaults to being empty unless --all or --include-history is supplied. This is useful if you want to track what bug fixes, etc, might be included in the released version. But for many projects it is more information than needed.
- file\_revisions: A dictionary listing the last-modified revision for all files in the project. This can be used similarly to how \$Id\$ keywords are used in CVS-controlled files. The last modified date can be determined by looking in the revisions map. This is also empty by default, and enabled only by --all or --include-file-revisions.

### Getting version info in other formats

Bazaar supports a template-based method for getting version information in arbitrary formats. The --custom option to version-info can be used by providing a --template argument that contains variables that will be expanded based on the status of the working tree. For example, to generate a C header file with a formatted string containing the current revision number:

```
bzr version-info --custom \
     --template="#define VERSION_INFO \"Project 1.2.3 (r{revno}
     > version_info.h
```

where the {revno} will be replaced by the revision number of the working tree. (If the example above doesn't work on your OS, try entering the command all on one line.) For more information on the variables that can be used in templates, see Version Info in the Bazaar User Reference.

Predefined formats for dumping version information in specific languages are currently in development. Please contact us on the mailing list about your requirements in this area.

#### Check clean

Most information about the contents of the project can be cheaply determined by just reading the revision entry. However, it can be useful to know if the working tree was completely up-to-date when it was packaged, or if there was a local modification. By supplying either --all or --check-clean, bzr will inspect the working tree, and set the clean flag in version\_info, as well as set entries in file\_revisions as modified where appropriate.

# BzrTools

### Overview

BzrTools is a collection of useful enhancements to Bazaar. For installation instructions, see the BzrTools home page: http://bazaar-vcs.org/BzrTools. Here is a sample of the frequently used commands it provides.

## shell

bzr shell starts up a command interpreter than understands Bazaar commands natively. This has several advantages:

- There's no need to type bzr at the front of every command.
- Intelligent auto-completion is provided.
- Commands run slightly faster as there's no need to load Bazaar's libraries each time.

## cdiff

bzr cdiff provides a colored version of bzr diff output. On GNU/Linux, UNIX and OS X, this is often used like this:

bzr cdiff | less -R

## bzr-svn

#### Overview

bzr-svn lets developers use Bazaar as their VCS client on projects still using a central Subversion repository. Access to Subversion repositories is largely transparent, i.e. you can use most bzr commands directly on Subversion repositories exactly the same as if you were using bzr on native Bazaar branches.

Many bzr-svn users create a local mirror of the central Subversion trunk, work in local feature branches, and submit their overall change back to Subversion when it is ready to go. This lets them gain many of the advantages of distributed VCS tools without interrupting existing team-wide processes and tool integration hooks currently built on top of Subversion. Indeed, this is a common interim step for teams looking to adopt Bazaar but who are unable to do so yet for timing or non-technical reasons.

For installation instructions, see the bzr-svn home page: http://bazaar-vcs.org/BzrForeignBranches/Subversion.

### A simple example

Here's a simple example of how you can use bzr-svn to hack on a GNOME project like **beagle**. Firstly, setup a local shared repository for storing your branches in and checkout the trunk:

```
bzr init-repo beagle-repo
cd beagle-repo
bzr checkout svn+ssh://svn.gnome.org/svn/beagle/trunk beagle-tr
```

Next, create a feature branch and hack away:

```
bzr branch beagle-trunk beagle-feature1
cd beagle-feature1
(hack, hack, hack)
bzr commit -m "blah blah blah"
(hack, hack, hack)
bzr commit -m "blah blah blah"
```

When the feature is cooked, refresh your trunk mirror and merge your change:

```
cd ../beagle-trunk
bzr update
bzr merge ../beagle-feature1
bzr commit -m "Complete comment for SVN commit"
```

As your trunk mirror is a checkout, committing to it implicitly commits to the real Subversion trunk. That's it!

## Using a central repository mirror

For large projects, it often makes sense to tweak the recipe given above. In particular, the initial checkout can get quite slow so you may wish to import the Subversion repository into a Bazaar one once and for all for your project, and then branch from that native Bazaar repository instead. bzr-svn provides the svn-import command for doing this repository-to-repository conversion. Here's an example of how to use it:

```
bzr svn-import svn+ssh://svn.gnome.org/svn/beagle
```

Here's the recipe from above updated to use a central Bazaar mirror:

```
bzr init-repo beagle-repo
cd beagle-repo
bzr branch bzr+ssh://bzr.gnome.org/beagle.bzr/trunk beagle-trun
bzr branch beagle-trunk beagle-feature1
cd beagle-feature1
(hack, hack, hack)
bzr commit -m "blah blah blah"
(hack, hack, hack)
bzr commit -m "blah blah blah"
cd ../beagle-trunk
bzr pull
bzr merge ../beagle-feature1
bzr commit -m "Complete comment for SVN commit"
bzr push
```

In this case, committing to the trunk only commits the merge locally. To commit back to the master Subversion trunk, an additional command (bzr push) is required.

Note: You'll need to give pull and push the relevant URLs the first time you use those commands in the trunk branch. After that, bzr remembers them.

The final piece of the puzzle in this setup is to put scripts in place to keep the central Bazaar mirror synchronized with the Subversion one. This can be done by adding a cron job, using a Subversion hook, or whatever makes sense in your environment.

#### Limitations of bzr-svn

Bazaar and Subversion are different tools with different capabilities so there will always be some limited interoperability issues. Here are some examples current as of bzr-svn 0.5.4:

- Bazaar doesn't support versioned properties
- Bazaar doesn't support tracking of file copies.

See the bzr-svn web page, http://bazaar-vcs.org/BzrForeignBranches/Subversion, for the current list of constraints.

# Web browsing

#### Overview

There are a range of options available for providing a web view of a Bazaar repository, the main one being Loggerhead. The homepage of Loggerhead can be found at https://launchpad.net/loggerhead.

A list of alternative web viewers including download links can be found on http://bazaar-vcs.org/WebInterface.

Note: If your project is hosted or mirrored on Launchpad, Loggerhead code browsing is provided as part of the service.

## Bug trackers

Bazaar has a facility that allows you to associate a commit with a bug in the project's bug tracker. Other tools (or hooks) can then use this information to generate hyperlinks between the commit and the bug, or to automatically mark the bug closed in the branches that contain the commit.

# Associating commits and bugs

When you make a commit, you can associate it with a bug by using the --fixes option of commit. For example:

```
$ bzr commit --fixes lp:12345 -m "Properly close the connection
▶1
```

This records metadata in Bazaar linking the commit with bug 12345 in Launchpad. If you use a different bug tracker, it can be given its own tracker code (instead of 1p) and used instead. For details on how to configure this for Bugzilla, Trac, Roundup and other bug/issue trackers, refer to Bug Tracker Settings in the Bazaar User Reference.

# Metadata recording vs bug tracker updating

Recording metadata about bugs fixed at commit time is only one of the features needed for complete bug tracker integration. As Bazaar is a distributed VCS, users may be offline while committing so accessing the bug tracker itself at that time may not be possible. Instead, it is recommended that a hook be installed to update the bug tracker when changes are pushed to a central location appropriate for your project's workflow.

Note: This second processing stage is part of the integration provided by Launchpad when it scans external or hosted branches.

# Making corrections

This method of associating revisions and bugs does have some limitations. The first is that the association can only be made at commit time. This means that if you forget to make the association when you commit, or the bug is reported after you fix it, you generally cannot go back and add the link later.

Related to this is the fact that the association is immutable. If a bug is marked as fixed by one commit but that revision does not fully solve the bug, or there is a later regression, you cannot go back and remove the link.

Of course, bzr uncommit can always be used to undo the last commit in order to make it again with the correct options. This is commonly done to correct a bad commit message and it equally applies to correcting metadata recorded (via --fixes for example) on the last commit.

Note: uncommit is best done before incorrect revisions become public.

# Specifying revisions

# Revision identifiers and ranges

Bazaar has a very expressive way to specify a revision or a range of revisions. To specify a range of revisions, the upper and lower bounds are separated by the ... symbol. For example:

```
$ bzr log -r 1..4
```

You can omit one bound like:

```
$ bzr log -r 1..
$ bzr log -r ..4
```

Some commands take only one revision, not a range. For example:

```
$ bzr cat -r 42 foo.c
```

In other cases, a range is required but you want the length of the range to be one. For commands where this is relevant, the -c option is used like this:

```
$ bzr diff -c 42
```

# Available revision identifiers

The revision, or the bounds of the range, can be given using different format specifications as shown below.

argument type	description
number	revision number
revno:number	revision number
last:number	negative revision number
guid	globally unique revision id
revid:guid	globally unique revision id
before:rev	leftmost parent of "rev"
date-value	first entry after a given date
date:date-value	first entry after a given date
tag-name	revision matching a given tag
tag:tag-name	revision matching a given tag
ancestor:path	last merged revision from a branch
branch:path	latest revision on another branch
submit:path	common ancestor with submit branch

A brief introduction to some of these formats is given below. For complete details, see Revision Identifiers in the Bazaar User Reference.

# **Numbers**

Positive numbers denote revision numbers in the current branch. Revision numbers are labelled as "revno" in the output of bzr log. To display the log for the first ten revisions:

Negative numbers count from the latest revision, -1 is the last committed revision.

To display the log for the last ten revisions:

```
$ bzr log -r -10..
```

# revid

**revid** allows specifying a an internal revision ID, as shown by bzr log --show-ids and some other commands.

For example:

## before

#### before

"rev" specifies the leftmost parent of "rev", that is the revision that appears before "rev" in the revision history, or the revision that was current when "rev" was committed.

"rev" can be any revision specifier and may be chained.

For example:

```
$ bzr log -r before:before:4
...
revno: 2
...
```

### date

#### date

"value" matches the first history entry after a given date, either at midnight or at a specified time.

Legal values are:

- yesterday
- today
- tomorrow
- A YYYY-MM-DD format date.
- A YYYY-MM-DD,HH:MM:SS format date/time, seconds are optional (note the comma)

The proper way of saying "give me all the log entries for today" is:

```
$ bzr log -r date:yesterday..date:today
```

### **Ancestor**

#### ancestor:path

specifies the common ancestor between the current branch and a different branch. This is the same ancestor that would be used for merging purposes.

path may be the URL of a remote branch, or the file path to a local branch.

For example, to see what changes were made on a branch since it was forked off ../parent:

```
$ bzr diff -r ancestor:../parent
```

## **Branch**

#### branch

path specifies the latest revision in another branch.

path may be the URL of a remote branch, or the file path to a local branch.

For example, to get the differences between this and another branch:

\$ bzr diff -r branch:http://example.com/bzr/foo.dev

Home | Documentation | Table of Contents (2.2b1) » Bazaar User Guide » previous | next

# Organizing your workspace

# Common workspace layouts

The best way for a Bazaar user to organize their workspace for a project depends on numerous factors including:

- user role: project owner vs core developer vs casual contributor
- workflows: particularly the workflow the project encourages/mandates for making contributions
- size: large projects have different resource requirements to small ones.

There are at least 4 common ways of organizing one's workspace:

- lightweight checkout
- standalone tree
- feature branches
- switchable sandbox.

A brief description of each layout follows.

# Lightweight checkout

In this layout, the working tree is local and the branch is remote. This is the standard layout used by CVS and Subversion: it's simple and well understood.

#### To set up:

```
bzr checkout --lightweight URL project cd project
```

#### To work:

```
(make changes)
bzr commit
(make changes)
bzr commit
```

Note that each commit implicitly publishes the change to everyone else working from that branch. However, you need to be up to date with changes in the remote branch for the commit to succeed. To grab the latest code and merge it with your changes, if any:

bzr update

# Standalone tree

In this layout, the working tree & branch are in the one place. Unless a shared repository exists in a higher level directory, the repository is located in that same place as well. This is the default layout in Bazaar and it's great for small to moderately sized projects.

#### To set up:

```
bzr branch URL project
cd project
```

#### To work:

```
(make changes)
bzr commit
(make changes)
bzr commit
```

To publish changes to a central location:

```
bzr push [URL]
```

The URL for push is only required the first time.

If the central location has, in the meantime, received changes from other users, then you'll need to merge those changes into your local branch before you try to push again:

```
bzr merge
(resolve conflicts)
bzr commit
```

As an alternative, a checkout can be used. Like a branch, a checkout has a full copy of the history stored locally but the local branch is bound to the remote location so that commits are published to both

locations at once.

Note: A checkout is actually smarter than a local commit followed by a push. In particular, a checkout wil commit to the remote location first and only commit locally if the remote commit succeeds.

# Feature branches

In this layout, there are multiple branches/trees, typically sharing a repository. One branch is kept as a mirror of "trunk" and each unit-of-work (i.e. bug-fix or enhancement) gets its own "feature branch". This layout is ideal for most projects, particularly moderately sized ones.

#### To set up:

```
bzr init-repo project
cd project
bzr branch URL trunk
```

#### To start a feature branch:

```
bzr branch trunk featureX
cd featureX
```

#### To work:

```
(make changes)
bzr commit
(make changes)
bzr commit
```

To publish changes to a mailing list for review & approval:

```
bzr send
```

To publish changes to a public branch (that can then be registered as a Launchpad merge request, say):

```
bzr push [URL]
```

As a variation, the trunk can be created as a checkout. If you have

commit privileges on trunk, that lets you merge into trunk and the commit of the merge will implicitly publish your change. Alternatively, if the trunk URL is read-only (e.g. a http address), that prevents accidental submission this way - ideal if the project workflow uses an automated gatekeeper like PQM, say.

# Local sandbox

This layout is very similar to the feature branches layout except that the feature branches share a single working tree rather than having one each. This is similar to git's default layout and it's useful for projects with really large trees (> 10000 files say) or for projects with lots of build artifacts (like .o or .class files).

#### To set up:

```
bzr init-repo --no-trees project
cd project
bzr branch URL trunk
bzr checkout --lightweight trunk sandbox
cd sandbox
```

While you *could* start making changes in sandbox now, committing while the sandbox is pointing to the trunk would mean that trunk is no longer a mirror of the upstream URL (well unless the trunk is a checkout). Therefore, you usually want to immediately create a feature branch and switch your sandbox to it like this:

```
bzr branch ../trunk ../featureX
bzr switch ../featureX
```

The processes for making changes and submitting them are otherwise pretty much the same as those used for feature branches.

# **Advanced layouts**

If you wish, you can put together your own layout based on how **you** like things organized. See Advanced shared repository layouts for examples and inspiration.

Home | Documentation | Table of Contents (2.2b1) » Bazaar User Guide » previous | next

# Advanced shared repository layouts

Bazaar is designed to give you flexibility in how you layout branches inside a shared repository. This flexibility allows users to tailor Bazaar to their workflow, but it also leads to questions about what is a "good" layout. We present some alternatives and give some discussion about the benefits of each.

One key point which should be mentioned is that any good layout should somehow highlight what branch a "general" user should grab. In SVN this is deemed the "trunk/" branch, and in most of the layouts this naming convention is preserved. Some would call this "mainline" or "dev", and people from CVS often refer to this as "HEAD".

# "SVN-Style" (trunk/, branches/)

Most people coming from SVN will be familiar with their "standard" project layout. Which is to layout the repository as:

```
repository/ # Overall repository
+- trunk/ # The mainline of development
+- branches/ # A container directory
| +- foo/ # Branch for developing feature foo
| ...
+- tags/ # Container directory
+- release-X # A branch specific to mark a given release v
...
```

With Bazaar, that is a perfectly reasonable layout. It has the benefit of being familiar to people coming from SVN, and making it clear where the development focus is.

When you have multiple projects in the same repository, the SVN layout is a little unclear what to do.

## project/trunk

The preferred method for SVN seems to be to give each project a top level directory for a layout like:

```
repository/ # Overall repository
+- project1/ # A container directory
| +- trunk/ # The mainline of development of project
| +- branches/ # A container directory
| +- foo/ # Branch for developing feature foo of p
| ...
| +- project2/ # Container for project2
+- trunk/ # Mainline for project2
+- branches/ # Container for project2 branches
```

This also works with Bazaar. However, with Bazaar repositories are cheap to create (a simple bzr init-repo away), and their primary benefit is when the branches share a common ancestry.

So the preferred way for Bazaar would be:

## trunk/project

There are also a few projects who use this layout in SVN:

```
repository/ # Overall repository
+- trunk/ # A container directory
| +- project1 # Mainline for project 1
| +- project2 # Mainline for project 2
| ...
| +- branches/ # Container
| +- project1/ # Container (?)
| +- foo # Branch 'foo' of project1
| +- project2/ +- bar # Branch 'bar' of project2
```

#### A slight variant is:

```
repository/ # Overall repository
+- trunk/ # A container directory
| +- project1 # Mainline for project 1
| +- project2 # Mainline for project 2
| ...
+- branches/ # Container
```

```
+- project1-foo/ # Branch 'foo' of project1
+- project2-bar/ # Branch 'bar' of project2
```

I believe the reason for this in SVN, is so that someone can checkout all of "trunk/" and get the all the mainlines for all projects.

This layout can be used for Bazaar, but it is not generally recommended.

- 1. bzr branch/checkout/get is a single branch at a time. So you don't get the benefit of getting all mainlines with a single command. [1]
- 2. It is less obvious of whether repository/trunk/foo is the trunk of project foo or it is just the foo directory in the trunk branch. Some of this confusion is due to SVN, because it uses the same "namespace" for files in a project that it uses for branches of a project. In Bazaar, there is a clear distinction of what files make up a project, versus the location of the Branch. (After all, there is only one .bzr/directory per branch, versus many .svn/directories in the checkout).

Note: NestedTreeSupport can provide a way to create "metaprojects" which aggregate multiple projects regardless of the repository layout. Letting you bzr checkout one project, and have it grab all the necessary sub-projects.

# Nested Style (project/branch/sub-branch/)

Another style with Bazaar, which is not generally possible in SVN is to have branches nested within each-other. This is possible because Bazaar supports (and recommends) creating repositories with no working trees (--no-trees). With a --no-trees repository, because the working files are not intermixed with your branch locations, you are free to put a branch in whatever namespace you want.

#### One possibility is:

```
project/  # The overall repository, *and* the projec  # Developer Joe's primary branch of develo  # Developer Joe's feature1 development bra  # A staging branch for Joe to develop feat  # Joe's feature2 development branch  # Barry's development branch  # Barry's development branch  # The overall repository, *and* the projec  # Developer Joe's primary branch of development branch  # A staging branch for Joe to develop feat  # Joe's feature2 development branch  # Barry's development branch  # Barry's development branch  # The overall repository, *and* the projec  # Development branch of development branch  # A staging branch for Joe to develop feat  # Joe's feature2 development branch  # Barry's development branch  # Barry's development branch  # The overall repository, *and* the projec  # Developer Joe's primary branch of development branch  # Developer Joe's feature1 development branch  # Developer Joe's feature2  # Developer Joe's feature2  # Developer Joe's feature3  # Developer Joe's fea
```

The idea with this layout is that you are creating a hierarchical layout for branches. Where changes generally flow upwards in the namespace. It also gives people a little corner of the namespace to work on their stuff. One nice feature of this layout, is it makes branching "cheaper" because it gives you a place to put all the mini branches without cluttering up the global branches/ namespace.

The other power of this is that you don't have to repeat yourself when specifying more detail in the branch name.

### For example compare:

```
bzr branch http://host/repository/project/branches/joe-feature-
```

Versus:

```
bzr branch http://host/project/joe/foo/bugfix-10
```

Also, if you list the repository/project/branches/ directory you might see something like:

```
barry-feature-bar/
barry-bugfix-10/
barry-bugfix-12/
joe-bugfix-10/
joe-bugfix-13/
joe-frizban/
```

Versus having these broken out by developer. If the number of branches are small, branches/ has the nice advantage of being able to see all branches in a single view. If the number of branches is large, branches/ has the distinct disadvantage of seeing all the branches in a single view (it becomes difficult to find the branch you are interested in, when there are 100 branches to look through).

Nested branching seems to scale better to larger number of branches. However, each individual branch is less discoverable. (eg. "Is Joe working on bugfix 10 in his feature foo branch, or his feature bar branch?")

One other small advantage is that you can do something like:

```
bzr branch http://host/project/release/1/1/1
or
bzr branch http://host/project/release/1/1/2
```

To indicate release 1.1.1 and 1.1.2. This again depends on how many releases you have and whether the gain of splitting things up outweighs the ability to see more at a glance.

# Sorted by Status (dev/, merged/, experimental/)

One other way to break up branches is to sort them by their current status. So you would end up with a layout something like:

```
project/ # Overall layout
+- trunk/ # The development focus branch
+- dev/ # Container directory for in-progress wo
| +- joe-feature1 # Joe's current feature-1 branch
| +- barry-bugfix10 # Barry's work for bugfix 10
| ...
+- merged/ # Container indicating these branches ha
| +- bugfix-12 # Bugfix which has already been merged.
+- abandonded/ # Branches which are considered 'dead-en
```

This has a couple benefits and drawbacks. It lets you see what branches are actively being developed on, which is usually only a small number, versus the total number of branches ever created. Old branches are not lost (versus deleting them), but they are "filed away", such that the more likely you are to want a branch the easier it is to find. (Conversely, older branches are likely to be harder to find).

The biggest disadvantage with this layout, is that branches move Which that if around. means someone is following project/dev/new-feature branch, when it gets merged into trunk/ suddenly bzr pull doesn't mirror the branch for them anymore because the branch is now at project/merged/new-feature. There are a couple ways around this. One is to use HTTP redirects to point people requesting the old branch to the new branch. bzr >= 0.15 will let users know that http://old/path redirects to http://new/path. However, this doesn't help if people are accessing a branch through methods other than HTTP (SFTP, local filesystem, etc).

It would also be possible to use a symlink for temporary redirecting (as long as the symlink is within the repository it should cause little trouble). However eventually you want to remove the symlink, or you don't get the clutter reduction benefit. Another possibility instead of a symlink is to use a BranchReference. It is currently difficult to create these through the bzr command line, but if people find them useful that could be changed. This is actually how Launchpad allows you to bzr checkout https://launchpad.net/bzr. Effectively a BranchReference is a symlink, but it allows you to reference any other URL. If it is extended to support relative references, it would even work over http, sftp, and local paths.

# Sorted by date/release/etc (2006-06/, 2006-07/, 0.8/, 0.9)

Another method of allowing some scalability while also allowing the browsing of "current" branches. Basically, this works on the assumption that actively developed branches will be "new" branches, and older branches are either merged or abandoned.

Basically the date layout looks something like:

```
project/ # Overall project repository
+- trunk/ # General mainline
+- 2006-06/ # containing directory for branches cre
| +- feature1/ # Branch of "project" for "feature1"
| +- feature2/ # Branch of "project" for "feature2"
+- 2005-05/ # Containing directory for branches cre
+- feature3/
...
```

This answers the question "Where should I put my new branch?" very quickly. If a feature is developed for a long time, it is even reasonable to copy a branch into the newest date, and continue working on it there. Finding an active branch generally means going to the newest date, and going backwards from there. (A small disadvantage is that most directory listings sort oldest to the top, which may mean more scrolling). If you don't copy old branches to newer locations, it also has the disadvantage that searching for a branch may take a while.

Another variant is by release target:

```
project/ # Overall repository
+- trunk/ # Mainline development branch
+- releases/ # Container for release branches
| +- 0.8/ # The branch for release 0.8
| +- 0.9/ # The branch for release 0.9
```

```
+- 0.8/ # Container for branches targeting release 0.
| +- feature1/ # Branch for "feature1" which is intended to
| +- feature2/ # Branch for "feature2" which is targeted for
+- 0.9/
+- feature3/ # Branch for "feature3", targeted for release
```

Some possible variants include having the 0.9 directory imply that it is branched *from* 0.9 rather than *for* 0.9, or having the 0.8/release as the official release 0.8 branch.

The general idea is that by targeting a release, you can look at what branches are waiting to be merged. It doesn't necessarily give you a good idea of what the state of the branch (is it in development or finished awaiting review). It also has a history-hiding effect, and otherwise has the same benefits and deficits as a date-based sorting.

# Simple developer naming (project/joe/foo, project/barry/bar)

Another possibly layout is to give each developer a directory, and then have a single sub-directory for branches. Something like:

```
project/ # Overall repository
+- trunk/ # Mainline branch
+- joe/ # A container for Joe's branches
| +- foo/ # Joe's "foo" branch of "project"
+- barry/
+- bar/ # Barry's "bar" branch of "project"
```

The idea is that no branch is "nested" underneath another one, just that each developer has his/her branches grouped together.

A variant which is used by Launchpad is:

This lets you easily browse what each developer is working on. Focus branches are kept in a "group" directory, which lets you see what branches the "group" is working on.

This keeps different people's work separated from each-other, but also makes it hard to find "all branches for project X". Launchpad compensates for this by providing a nice web interface with a database back end, which allows a "view" to be put on top of this layout. This is closer to the model of people's home pages, where each person has a "~/public\_html" directory where they can publish their own web-pages. In general, though, when you are creating a shared repository for centralization of a project, you don't want to split it up by person and then project. Usually you would want to split it up by project and then by person.

# Summary

In the end, no single naming scheme will work for everyone. It depends a lot on the number of developers, how often you create a new branch, what sort of lifecycles your branches go through. Some questions to ask yourself:

- 1. Do you create a few long-lived branches, or do you create lots of "mini" feature branches (Along with this is: Would you *like* to create lots of mini feature branches, but can't because they are a pain in your current VCS?)
- 2. Are you a single developer, or a large team?
- 3. If a team, do you plan on generally having everyone working on the same branch at the same time? Or will you have a "stable" branch that people are expected to track.

# Configuring email

# Why set up an email address with Bazaar?

Bazaar stores the specified email address in revisions when they're created so that people can tell who committed which revisions. The email addresses are not verified, therefore they could be bogus, so you have to trust the people involved in your project. Additionally, the email address in a revision gives others a way to contact the author of a revision for credit and/or blame. :)

# How to set up your email address

Bazaar will try to guess an email address based on your username and the hostname if none is set. This will probably not be what you want, so three ways exist to tell Bazaar what email to use:

You can set your email in one of several configuration files. Like other configuration values, you can set it in bazaar.conf as a general setting. If you want to override the value for a particular branch, or set of branches, you can use locations.conf. bzr/branch/branch.conf will also work, but will cause all commits to that branch to use the same email address, even if someone else does them.

#### The order of precedence is

- 1. If the BZR\_EMAIL environment variable is set.
- 2. If an email is set for your current branch in the locations.conf file.
- 3. If an email is set four your current branch in the .bzr/branch/branch.conf file.
- 4. If an email is set in the bazaar.conf default configuration file.
- 5. If the *EMAIL* environment variable is set.
- 6. Bazaar will try to guess based on your username and the hostname.

To check on what Bazaar thinks your current email is, use the whoami ("who am i?") command:

# Setting email via the 'whoami' command

You can use the whoami command to set your email globally:

```
% bzr whoami "Joe Cool <joe@example.com>"
```

or only for the current branch:

```
% bzr whoami --branch "Joe Cool <joe@example.com>"
```

These modify your global bazaar.conf or branch branch.conf file, respectively.

# Setting email via default configuration file

To use the default ini file, create or edit the bazaar.conf file (in ~/.bazaar/ on Linux and in %APPDATA%\bazaar\2.0\ in Windows) and set an email address as shown below. Please note that the word DEFAULT is case sensitive, and must be in upper-case.

```
[DEFAULT]
email=Your Name <name@isp.com>
```

For more information on the ini file format, see Configuration Settings in the Bazaar User Reference.

# Setting email on a per-branch basis

The second approach is to set email on a branch by branch basis by using the locations.conf configuration file like this:

[/some/branch/location]
email=Your Name <name@other-isp.com>

This will set your email address in the branch at /some/branch/location, overriding the default specified in the bazaar.conf above.

# Setting email via environment variable

The final method Bazaar will use is checking for the BZR\_EMAIL and EMAIL environment variables. Generally, you would use this method to override the email in a script context. If you would like to set a general default, then please see the ini methods above.

## Concerns about spam

Some people want to avoid sharing their email address so as not to get spam. Bazaar will never disclose your email address, unless you publish a branch or changeset in a public location. It's recommended that you *do* use a real address, so that people can contact you about your work, but it's not required. You can use an address which is obfuscated, which bounces, or which goes through an anti-spam service such as *spamgourmet.com*.

Home | Documentation | Table of Contents (2.2b1) » Bazaar User Guide » previous | next

# Serving Bazaar with Apache

This document describes one way to set up a Bazaar HTTP smart server, using Apache 2.0 and FastCGI or mod\_python or mod\_wsgi.

For more information on the smart server, and other ways to configure it see the main smart server documentation.

# Example

You webserver already have a publishing /srv/example.com/www/code as http://example.com/code/... with branches and plain HTTP. It contains bzr directories like /srv/example.com/www/code/branch-one and /srv/example.com/www/code/my-repo/branch-two. You want to provide read-only smart server access to these directories in addition to the existing HTTP access.

## Configuring Apache 2.0

#### **FastCGI**

First, configure mod\_fastcgi, e.g. by adding lines like these to your httpd.conf:

```
LoadModule fastcgi_module /usr/lib/apache2/modules/mod_fastcgi.
FastCgiIpcDir /var/lib/apache2/fastcgi

| |
```

In our example, we're already serving /srv/example.com/www/code at http://example.com/code, so our existing Apache configuration would look like:

```
Alias /code /srv/example.com/www/code

<Directory /srv/example.com/www/code>

    Options Indexes

# ...

</Directory>
```

We need to change it to handle all requests for URLs ending in .bzr/smart. It will look like:

```
Alias /code /srv/example.com/www/code
<Directory /srv/example.com/www/code>
    Options Indexes FollowSymLinks
    RewriteEngine On
    RewriteBase /code
    RewriteRule ^(.*/|)\.bzr/smart$ /srv/example.com/scripts/bz
</Directory>

# bzr-smart.fcgi isn't under the DocumentRoot, so Alias it into
# namespace so it can be executed.
Alias /srv/example.com/scripts/bzr-smart.fcgi /srv/example.com/
<Directory /srv/example.com/scripts>
    Options ExecCGI
    <Files bzr-smart.fcgi>
        SetHandler fastcgi-script
```

```
</Files>
</Directory>
```

This instructs Apache to hand requests for any URL ending with /.bzr/smart inside /code to a Bazaar smart server via FastCGI.

Refer to the mod\_rewrite and mod\_fastcgi documentation for further information.

#### mod python

First, configure mod\_python, e.g. by adding lines like these to your httpd.conf:

```
LoadModule python_module /usr/lib/apache2/modules/mod_python.so
```

Define the rewrite rules with mod\_rewrite the same way as for FastCGI, except change:

```
RewriteRule ^(.*/|)\.bzr/smart$ /srv/example.com/scripts/bzr-sm
```

to:

```
RewriteRule ^(.*/|)\.bzr/smart$ /srv/example.com/scripts/bzr-sm
```

Like with mod\_fastcgi, we also define how our script is to be handled:

```
</Directory>
```

This instructs Apache to hand requests for any URL ending with /.bzr/smart inside /code to a Bazaar smart server via mod python.

NOTE: If you don't have bzrlib in your PATH, you will be need to change the following line:

Refer to the mod\_python documentation for further information.

#### mod wsgi

First, configure mod\_wsgi, e.g. enabling the mod with a2enmod wsgi. We need to change it to handle all requests for URLs ending in .bzr/smart. It will look like:

This instructs Apache to hand requests for any URL ending with /.bzr/smart inside /code to a Bazaar smart server via WSGI, and any other URL inside /code to be served directly by Apache. Refer to the  ${\hbox{mod\_wsgi}}$  documentation for further information.

# **Configuring Bazaar**

#### **FastCGI**

We've configured Apache to run the smart server at /srv/example.com/scripts/bzr-smart.fcgi. This is just a simple script we need to write to configure a smart server, and glue it to the FastCGI gateway. Here's what it looks like:

```
import fcgi
from bzrlib.transport.http import wsgi

smart_server_app = wsgi.make_app(
    root='/srv/example.com/www/code',
    prefix='/code/',
    path_var='REQUEST_URI',
    readonly=True,
    load_plugins=True,
    enable_logging=True)

fcgi.WSGIServer(smart_server_app).run()
```

The *fcgi* module can be found at http://svn.saddi.com/py-lib/trunk/fcgi.py. It is part of flup.

### mod python

We've configured Apache to run the smart server at /srv/example.com/scripts/bzr-smart.py. This is just a simple script we need to write to configure a smart server, and glue it to the mod\_python gateway. Here's what it looks like:

```
import modpywsgi
from bzrlib.transport.http import wsgi
smart_server_app = wsgi.make_app(
    root='/srv/example.com/www/code',
```

```
prefix='/code/',
    path_var='REQUEST_URI',
    readonly=True,
    load_plugins=True,
    enable_logging=True)

def handler(request):
    """Handle a single request."""
    wsgi_server = modpywsgi.WSGIServer(smart_server_app)
    return wsgi_server.run(request)
```

The *modpywsgi* module can be found at http://ice.usq.edu.au/svn/ice/trunk/apps/ice-server/modpywsgi.py. It was part of pocoo. You sould make sure you place modpywsgi.py in the same directory as bzr-smart.py (ie. /srv/example.com/scripts/).

#### mod\_wsgi

We've configured Apache to run the smart server at /srv/example.com/scripts/bzr.wsgi. This is just a simple script we need to write to configure a smart server, and glue it to the WSGI gateway. Here's what it looks like:

```
from bzrlib.transport.http import wsgi

def application(environ, start_response):
    app = wsgi.make_app(
        root="/srv/example.com/www/code/",
        prefix="/code",
        readonly=True,
        enable_logging=False)
    return app(environ, start_response)
```

# Clients

Now you can use bzr+http:// URLs or just http:// URLs, e.g.:

bzr log bzr+http://example.com/code/my-branch

Plain HTTP access should continue to work:

bzr log http://example.com/code/my-branch

### Advanced configuration

Because the Bazaar HTTP smart server is a WSGI application, it can be used with any 3rd-party WSGI middleware or server that conforms the WSGI standard. The only requirements are:

- to construct a *SmartWSGIApp*, you need to specify a **root transport** that it will serve.
- each request's *environ* dict must have a 'bzrlib.relpath' variable set.

The *make\_app* helper used in the example constructs a *SmartWSGIApp* with a transport based on the *root* path given to it, and calculates the 'bzrlib.relpath' for each request based on the *prefix* and *path\_var* arguments. In the example above, it will take the 'REQUEST\_URI' (which is set by Apache), strip the '/code/' prefix and the '/.bzr/smart' suffix, and set that as the 'bzrlib.relpath', so that a request for '/code/foo/bar/.bzr/smart' will result in a 'bzrlib.relpath' of 'foo/bzr'.

It's possible to configure a smart server for a non-local transport, or that does arbitrary path translations, etc, by constructing a *SmartWSGIApp* directly. Refer to the docstrings of *bzrlib.transport.http.wsgi* and the WSGI standard for further information.

### Pushing over the http smart server

It is possible to allow pushing data over the http smart server. The easiest way to do this, is to just supply readonly=False to the wsgi.make\_app() call. But be careful, because the smart protocol does not contain any Authentication. So if you enable write support, you will want to restrict access to .bzr/smart URLs to restrict who

can actually write data on your system, e.g. in apache it looks like:

```
<Location /code>
   AuthType Basic
   AuthName "example"
   AuthUserFile /srv/example.com/conf/auth.passwd
   <LimitExcept GET>
        Require valid-user
   </LimitExcept>
</Location>
```

At this time, it is not possible to allow some people to have read-only access and others to have read-write access to the same urls. Because at the HTTP layer (which is doing the Authenticating), everything is just a POST request. However, it would certainly be possible to have HTTPS require authentication and use a writable server, and plain HTTP allow read-only access.

If bzr gives an error like this when accessing your HTTPS site:

```
bzr: ERROR: Connection error: curl connection error (server cer
CAfile:/etc/ssl/certs/ca-certificates.crt CRLfile: none)
```

You can workaround it by using <a href="https://https:

# Writing a plugin

# Introduction

Plugins are very similar to bzr core functionality. They can import anything in bzrlib. A plugin may simply override standard functionality, but most plugins supply new commands.

# Creating a new command

To create a command, make a new object that derives from bzrlib.commands.command, and name it cmd\_foo, where foo is the name of your command. If you create a command whose name contains an underscore, it will appear in the UI with the underscore turned into a hyphen. For example, cmd\_baz\_import will appear as baz-import. For examples of how to write commands, please see builtins.py.

Once you've created a command you must register the command with bzrlib.commands.register\_command(cmd\_foo). You must register the command when your file is imported, otherwise bzr will not see it.

# Specifying a plugin version number

Simply define version\_info to be a tuple defining the current version number of your plugin. eg. version\_info = (0, 9, 0) version\_info = (0, 9, 0, 'dev', 0)

# Plugin searching rules

Bzr will scan ~/.bazaar/plugins and bzrlib/plugins for plugins by default. You can override this with BZR\_PLUGIN\_PATH (see User Reference for details).

Plugins may be either modules or packages. If your plugin is a single file, you can structure it as a module. If it has multiple files, or if you want to distribute it as a bzr branch, you should structure it as a package, i.e. a directory with an <u>\_\_init\_\_.py</u> file.

## More information

Please feel free to contribute your plugin to BzrTools, if you think it would be useful to other people.

See the Bazaar Developer Guide for details on Bazaar's development guidelines and policies.

Home | Documentation | Table of Contents (2.2b1) » Bazaar User Guide » previous | next

# **Bazaar Tutorial**

### Introduction

If you are already familiar with decentralized version control, then please feel free to skip ahead to "Introducing Yourself to Bazaar". If, on the other hand, you are familiar with version control but not decentralized version control, then please start at "How DVCS is different." Otherwise, get some coffee or tea, get comfortable and get ready to catch up.

## The purpose of version control

Odds are that you have worked on some sort of textual data – the sources to a program, web sites or the config files that Unix system administrators have to deal with in /etc. The chances are also good that you have made some sort of mistake that you deeply regretted. Perhaps you deleted the configuration file for your mailserver or perhaps mauled the source code for a pet project. Whatever happened, you have just deleted important information that you would desperately like to get back. If this has ever happened to you, then you are probably ready for Bazaar.

Version control systems (which I'll henceforth call VCS) such as Bazaar give you the ability to track changes for a directory by turning it into something slightly more complicated than a directory that we call a **branch**. The branch not only stores how the directory looks right now, but also how it looked at various points in the past. Then, when you do something you wish you hadn't, you can restore the directory to the way it looked at some point in the past.

Version control systems give users the ability to save changes to a branch by "committing a **revision**". The revision created is essentially a summary of the changes that were made since the last time the tree was saved.

These revisions have other uses as well. For example, one can comment revisions to record what the recent set of changes meant by providing an optional log message. Real life log messages include things like "Fixed the web template to close the table" and "Added sftp suppport. Fixes #595"

We keep these logs so that if later there is some sort of problem with sftp, we can figure out when the problem probably happened.

### How DVCS is different

Many Version Control Systems (VCS) are stored on servers. If one wants to work on the code stored within a VCS, then one needs to connect to the server and "checkout" the code. Doing so gives one a directory in which a person can make changes and then commit. The VCS client then connects to the VCS server and stores the changes. This method is known as the centralized model.

The centralized model can have some drawbacks. A centralized VCS requires that one is able to connect to the server whenever one wants to do version control work. This can be a bit of a problem if your server is on some other machine on the internet and you are not. Or, worse yet, you **are** on the internet but the server is missing!

Decentralized Version Control Systems (which I'll call DVCS after this point) deal with this problem by keeping branches on the same machine as the client. In Bazaar's case, the branch is kept in the same place as the code that is being version controlled. This allows the user to save his changes (**commit**) whenever he wants – even if he is offline. The user only needs internet access when he wants to access the changes in someone else's branch that are somewhere else.

A common requirement that many people have is the need to keep track of the changes for a directory such as file and subdirectory changes. Performing this tracking by hand is a awkward process that over time becomes unwieldy. That is, until one considers version control tools such as Bazaar. These tools automate the process of storing data by creating a **revision** of the directory tree whenever the user asks.

Version control software such as Bazaar can do much more than just storage and performing undo. For example, with Bazaar a developer can take the modifications in one branch of software and apply them to a related branch – even if those changes exist in a branch owned by somebody else. This allows developers to cooperate without giving write access to the repository.

Bazaar remembers the "ancestry" of a revision: the previous revisions that it is based upon. A single revision may have more than one direct descendant, each with different changes, representing a divergence in the evolution of the tree. By branching, Bazaar allows multiple people to cooperate on the evolution of a project, without all needing to work in strict lock-step. Branching can be useful even for a single developer.

## Introducing yourself to Bazaar

Bazaar installs a single new command, **bzr**. Everything else is a subcommand of this. You can get some help with bzr help. Some arguments are grouped in topics: bzr help topics to see which topics are available.

One function of a version control system is to keep track of who changed what. In a decentralized system, that requires an identifier for each author that is globally unique. Most people already have one of these: an email address. Bazaar is smart enough to automatically generate an email address by looking up your username and hostname. If you don't like the guess that Bazaar makes, then three options exist:

1. Set an email address via bzr whoami. This is the simplest way.

To set a global identity, use:

```
% bzr whoami "Your Name <email@example.com>"
```

If you'd like to use a different address for a specific branch, enter the branch folder and use:

```
% bzr whoami --branch "Your Name <email@example.com>"
```

2. Setting the email address in the ~/.bazaar/bazaar.conf [1] by adding the following lines. Please note that [DEFAULT] is case sensitive:

```
[DEFAULT]
email=Your Name <email@isp.com>
```

As above, you can override this settings on a branch by branch

basis by creating a branch section in ~/.bazaar/locations.conf and adding the following lines:

[/the/path/to/the/branch]
email=Your Name <email@isp.com>

- 3. Overriding the two previous options by setting the global environment variable \$BZR\_EMAIL or \$EMAIL (\$BZR\_EMAIL will take precedence) to your full email address.
  - (1, 2) On Windows, the users configuration files can be found in the application data directory. So instead of
- [1] ~/.bazaar/branch.conf the configuration file can be found as: C:\Documents and Settings\<username>\Application
  Data\Bazaar\2.0\branch.conf. The same is true for locations.conf, ignore, and the plugins directory.

### Creating a branch

History is by default stored in the .bzr directory of the branch. In a future version of Bazaar, there will be a facility to store it in a separate repository, which may be remote.

We create a new branch by running bzr init in an existing directory:

```
% mkdir tutorial
% cd tutorial
% ls -a
./ ../
% pwd
/home/mbp/work/bzr.test/tutorial
%
% bzr init
% ls -aF
./ ../ .bzr/
%
```

As with CVS, there are three classes of file: unknown, ignored, and versioned. The **add** command makes a file versioned: that is, changes to it will be recorded by the system:

```
% echo 'hello world' > hello.txt
% bzr status
unknown:
   hello.txt
% bzr add hello.txt
added hello.txt
% bzr status
added:
   hello.txt
```

If you add the wrong file, simply use bzr remove to make it unversioned again. This does not delete the working copy in this case, though it may in others [2].

(1, 2) bzr remove will remove the working copy if it is currently

[2] versioned, but has no changes from the last committed version. You can force the file to always be kept with the --keep option to bzr remove, or force it to always be deleted with --force.

#### **Branch locations**

All history is stored in a branch, which is just an on-disk directory containing control files. By default there is no separate repository or database as used in svn or svk. You can choose to create a repository if you want to (see the bzr init-repo command). You may wish to do this if you have very large branches, or many branches of a moderately sized project.

You'll usually refer to branches on your computer's filesystem just by giving the name of the directory containing the branch. bzr also supports accessing branches over http and sftp, for example:

```
% bzr log http://bazaar-vcs.org/bzr/bzr.dev/
% bzr log sftp://bazaar-vcs.org/bzr/bzr.dev/
```

By installing bzr plugins you can also access branches using the rsync protocol.

See the Publishing your branch section for more about how to put your branch at a given location.

# Reviewing changes

Once you have completed some work, you will want to **commit** it to the version history. It is good to commit fairly often: whenever you get a new feature working, fix a bug, or improve some code or documentation. It's also a good practice to make sure that the code compiles and passes its test suite before committing, to make sure that every revision is a known-good state. You can also review your changes, to make sure you're committing what you intend to, and as a chance to rethink your work before you permanently record it.

Two bzr commands are particularly useful here: **status** and **diff**.

#### bzr status

The **status** command tells you what changes have been made to the working directory since the last revision:

```
% bzr status
modified:
   foo
```

bzr status hides "boring" files that are either unchanged or ignored. The status command can optionally be given the name of some files or directories to check.

#### bzr diff

The **diff** command shows the full text of changes to all files as a standard unified diff. This can be piped through many programs such as "patch", "diffstat", "filterdiff" and "colordiff":

```
% bzr diff
=== added file 'hello.txt'
```

```
--- hello.txt 1970-01-01 00:00:00 +0000
+++ hello.txt 2005-10-18 14:23:29 +0000
@@ -0,0 +1,1 @@
+hello world
```

With the <u>-r</u> option, the tree is compared to an earlier revision, or the differences between two versions are shown:

The --diff-options option causes bzr to run the external diff program, passing options. For example:

```
% bzr diff --diff-options --side-by-side foo
```

Some projects prefer patches to show a prefix at the start of the path for old and new files. The --prefix option can be used to provide such a prefix. As a shortcut, bzr diff -p1 produces a form that works with the command patch -p1.

# Committing changes

When the working tree state is satisfactory, it can be **committed** to the branch, creating a new revision holding a snapshot of that state.

#### bzr commit

The **commit** command takes a message describing the changes in the revision. It also records your userid, the current time and timezone, and the inventory and contents of the tree. The commit message is specified by the -m or --message option. You can enter a multi-line commit message; in most shells you can enter this just by leaving the quotes open at the end of the line.

```
% bzr commit -m "added my first file"
```

You can also use the **-F** option to take the message from a file. Some people like to make notes for a commit message while they work, then review the diff to make sure they did what they said they did. (This file can also be useful when you pick up your work after a break.)

#### Message from an editor

If you use neither the -m nor the -F option then bzr will open an editor for you to enter a message. The editor to run is controlled by your \$VISUAL or \$EDITOR environment variable, which can be overridden by the editor setting in ~/.bazaar/bazaar.conf; \$BZR\_EDITOR will override either of the above mentioned editor options. If you quit the editor without making any changes, the commit will be cancelled.

The file that is opened in the editor contains a horizontal line. The part of the file below this line is included for information only, and will not form part of the commit message. Below the separator is shown the list of files that are changed in the commit. You should write your message above the line, and then save the file and exit.

If you would like to see the diff that will be committed as you edit the message you can use the --show-diff option to commit. This will include the diff in the editor when it is opened, below the separator and the information about the files that will be committed. This means that you can read it as you write the message, but the diff itself wont be seen in the commit message when you have finished. If you would like parts to be included in the message you can copy and paste them above the separator.

#### Marking bugs as fixed

Many changes to a project are as a result of fixing bugs. Bazaar can keep metadata about bugs you fixed when you commit them. To do this you use the --fixes option. This option takes an argument that looks like this:

```
% bzr commit --fixes <tracker>:<id>
```

Where <tracker> is an identifier for a bug tracker and <id> is an identifier for a bug that is tracked in that bug tracker. <id> is usually a number. Bazaar already knows about a few popular bug trackers. They are bugs.launchpad.net, bugs.debian.org, and bugzilla.gnome.org. These trackers have their own identifiers: lp, deb, and gnome respectively. For example, if you made a change to fix the bug #1234 on bugs.launchpad.net, you would use the following command to commit your fix:

For more information on this topic or for information on how to configure other bug trackers please read Bug Tracker Settings.

#### Selective commit

If you give file or directory names on the commit command line then only the changes to those files will be committed. For example:

```
% bzr commit -m "documentation fix" commit.py
```

By default bzr always commits all changes to the tree, even if run from a subdirectory. To commit from only the current directory down, use:

```
% bzr commit .
```

# Removing uncommitted changes

If you've made some changes and don't want to keep them, use the **revert** command to go back to the previous head version. It's a good idea to use bzr diff first to see what will be removed. By default the revert command reverts the whole tree; if file or directory names are given then only those ones will be affected. bzr revert also clears the list of pending merges revisions.

# Ignoring files

#### The .bzrignore file

Many source trees contain some files that do not need to be versioned, such as editor backups, object or bytecode files, and built programs. You can simply not add them, but then they'll always crop up as unknown files. You can also tell bzr to ignore these files by adding them to a file called .bzrignore at the top of the tree.

This file contains a list of file wildcards (or "globs"), one per line. Typical contents are like this:

```
*.o
*~
*.tmp
*.py[co]
```

If a glob contains a slash, it is matched against the whole path from the top of the tree; otherwise it is matched against only the filename. So the previous example ignores files with extension .o in all subdirectories, but this example ignores only config.h at the top level and HTML files in doc/:

```
./config.h
doc/*.html
```

To get a list of which files are ignored and what pattern they matched, use bzr ignored:

```
% bzr ignored config.h ./config.h configure.in~ *~
```

It is OK to have either an ignore pattern match a versioned file, or to

add an ignored file. Ignore patterns have no effect on versioned files; they only determine whether unversioned files are reported as unknown or ignored.

The .bzrignore file should normally be versioned, so that new copies of the branch see the same patterns:

```
% bzr add .bzrignore
% bzr commit -m "Add ignore patterns"
```

#### bzr ignore

As an alternative to editing the .bzrignore file, you can use the bzrignore command. The bzr ignore command takes filenames and/or patterns as arguments and then adds them to the .bzrignore file. If a .bzrignore file does not exist the bzr ignore command will automatically create one for you, and implicitly add it to be versioned:

```
% bzr ignore tags
% bzr status
added:
   .bzrignore
```

Just like when editing the .bzrignore file on your own, you should commit the automatically created .bzrignore file:

```
% bzr commit -m "Added tags to ignore file"
```

### Global ignores

There are some ignored files which are not project specific, but more user specific. Things like editor temporary files, or personal temporary files. Rather than add these ignores to every project, bzr supports a global ignore file in ~/.bazaar/ignore [1]. It has the same

syntax as the per-project ignore file.

# Examining history

## bzr log

The bzr log command shows a list of previous revisions. The bzr log --forward command does the same in chronological order to get most recent revisions printed at last.

As with bzr diff, bzr log supports the -r argument:

```
% bzr log -r 1000..  # Revision 1000 and everything aft  
% bzr log -r ..1000  # Everything up to and including r  
% bzr log -r 1000..1100  # changes from 1000 to 1100  # The changes in only revision 100
```

# **Branch statistics**

The bzr info command shows some summary information about the working tree and the branch history.

# Versioning directories

bzr versions files and directories in a way that can keep track of renames and intelligently merge them:

```
% mkdir src
% echo 'int main() {}' > src/simple.c
% bzr add src
added src
added src/simple.c
% bzr status
added:
    src/
    src/simple.c
```

# Deleting and removing files

You can delete files or directories by just deleting them from the working directory. This is a bit different to CVS, which requires that you also do cvs remove.

bzr remove makes the file un-versioned, but may or may not delete the working copy [2]. This is useful when you add the wrong file, or decide that a file should actually not be versioned.

```
% rm -r src
% bzr remove -v hello.txt
? hello.txt
% bzr status
removed:
 hello.txt
 src/
 src/simple.c
unknown:
 hello.txt
```

If you remove the wrong file by accident, you can use bzr revert to restore it.

# Branching

Often rather than starting your own project, you will want to submit a change to an existing project. To do this, you'll need to get a copy of the existing branch. Because this new copy is potentially a new branch, the command is called **branch**:

```
% bzr branch http://bazaar-vcs.org/bzr/bzr.dev
% cd bzr.dev
```

This copies down the complete history of this branch, so we can do all operations on it locally: log, annotate, making and merging branches. There will be an option to get only part of the history if you wish.

You can also get a copy of an existing branch by copying its directory, expanding a tarball, or by a remote copy using something like rsync.

# Following upstream changes

You can stay up-to-date with the parent branch by "pulling" in their changes:

% bzr pull

After this change, the local directory will be a mirror of the source. This includes the "revision-history" - which is a list of the commits done in this branch, rather than merged from other branches.

This command only works if your local (destination) branch is either an older copy of the parent branch with no new commits of its own, or if the most recent commit in your local branch has been merged into the parent branch.

# Merging from related branches

If two branches have diverged (both have unique changes) then bzr merge is the appropriate command to use. Merge will automatically calculate the changes that exist in the branch you're merging from that are not in your branch and attempt to apply them in your branch.

```
% bzr merge URL
```

If there is a conflict during a merge, 3 files with the same basename are created. The filename of the common base is appended with ".BASE", the filename of the file containing your changes is appended with ".THIS" and the filename with the changes from the other tree is appended with ".OTHER". Using a program such as kdiff3, you can now comfortably merge them into one file. In order to commit you have to rename the merged file (".THIS") to the original file name. To complete the conflict resolution you must use the resolve command, which will remove the ".OTHER" and ".BASE" files. As long as there exist files with .BASE, .THIS or .OTHER the commit command will report an error.

```
% kdiff3 file.BASE file.OTHER file.THIS
% mv file.THIS file
% bzr resolve file
```

[TODO: explain conflict markers within files]

# Publishing your branch

You don't need a special server to publish a bzr branch, just a normal web server. Just mirror the files to your server, including the .bzr directory. One can push a branch (or the changes for a branch) by one of the following three methods:

The best method is to use bzr itself to do it.

```
% bzr push sftp://servername.com/path/to/directory
```

(The destination directory must already exist unless the -- create-prefix option is used.)

- Another option is the rspush plugin that comes with BzrTools, which uses rsync to push the changes to the revision history and the working tree.
- You can also copy the files around manually, by sending a tarball, or using rsync, or other related file transfer methods. This is usually less safe than using push, but may be faster or easier in some situations.

# Moving changes between trees

It happens to the best of us: sometimes you'll make changes in the wrong tree. Maybe because you've accidentally started work in the wrong directory, maybe because as you're working, the change turns out to be bigger than you expected, so you start a new branch for it.

To move your changes from one tree to another, use

```
% cd NEWDIR
% bzr merge --uncommitted OLDDIR
```

This will apply all of the uncommitted changes you made in OLDDIR to NEWDIR. It will not apply committed changes, even if they could be applied to NEWDIR with a regular merge. The changes will remain in OLDDIR, but you can use bzr revert OLDDIR to remove them, once you're satisfied with NEWDIR.

NEWDIR does not have to be a copy of OLDDIR, but they should be related. The more different they are, the greater the chance of conflicts.

Home | Documentation | Table of Contents (2.2b1) » Tutorials »

previous | next

# Using Bazaar with Launchpad

#### Motivation

#### Communities are different to teams

The team of people required to create the initial release of a piece of software may vary in size from one person to several thousand people. Depending on the requirements, the challenges involved, both technical and managerial, can be immense. As explained in the Bazaar User Guide, selecting "just right" processes and using tools like Bazaar to support matching workflows can greatly help.

Success with software though requires more than a great team - it requires a healthy, active *community*. This group is typically far larger than the team as it includes everyone interested in the software: the team, users, training partners, support partners, third-party developers and so on.

Great communities are well understood in the open source world. Their applicability extends well beyond that though: most successful commercial software vendors are well skilled at building and managing the communities that grow up around their flagship products.

Like great teams, great communities don't just happen. Good policies and guidelines are essential for fostering the right sort of behaviour and healthy relationships between participants. For a deeper look at this topic, see Karl Fogel's landmark book: Producing Open Source Software.

#### The need for Collaborative Development Environments

An intelligent toolset is also important for tracking and managing

community information and workflows. These tools are called Collaborative Development Environments (CDEs). These toolsets are typically web-based and manage things such as announcements, issues/bugs, questions and answers, downloads, documents and source code. Some examples of CDEs include Launchpad, SourceForge, java.net and SAP Community Network.

# Helping communities work with related communities

Many successful products have a huge number of downstream dependencies. In other words, a new challenge arises with success: dealing with other communities and understanding how your changes will impact them. This is most obvious for projects like:

- software languages, e.g. Python, PHP, Ruby, Java, Perl, etc.
- compilers, e.g. gcc, JDK, etc.
- libraries, e.g. zlib, openssl, etc.
- frameworks, e.g. Zope, Ruby on Rails, Spring, etc.

However it applies equally for popular applications on which add-ons are built, e.g. Firefox, Thunderbird, OpenOffice.org, Drupal, Wordpress, Joomla, etc.

Tools that assist communities work together to track and manage issues and fixes across community boundaries are required. These tools help people at both ends of the spectrum:

- users can report problems in their terms, e.g. rendering of image type X is broken in application Y on operating system Z
- developers can better appreciate the downstream impact of making a change or fix, e.g. fixing this bug in a graphics library will make the users of these 5 applications on these 10 operating systems happy.

People in the middle play the essential role of *joining the dots* and communicating up and down the line. In many cases, they may also fix the problem for end users, releasing a patch and pushing a suggested fix to the upstream development team. Keeping track of all that over time in a sustainable way is no easy task.

#### Launchpad: More development, less friction

As well as sponsoring Ubuntu and Bazaar development, Canonical provides Launchpad, https://launchpad.net, as a free service for the open source community. Launchpad is one of the most exciting CDEs around for several notable reasons:

- it models relationships between many of things tracked, e.g. source code branches can be associated with bug fixes
- as well are managing historical knowledge, it supports future development planning and tracking by providing features such as roadmaps, milestones and blueprints
- it provides translation tools and packaging services so that barriers are reduced for translators and testers wishing to join your community and help out
- it provides a nexus for different communities to work together on related issues and roadmaps.

In other words, Launchpad has been designed to help your community grow and to reduce the workflow friction both *within* your community and *between* communities. Ultimately, that means less time on mechanical tasks and more time for interesting development.

#### Bazaar: Launchpad's VCS client

This tutorial looks at how Bazaar and Launchpad can be used together and how they complement each other. It is important to remember that:

- 1. Bazaar can be used without Launchpad
- 2. Launchpad can be used without Bazaar.

By design though, their sum is greater than the individual parts.

# Finding and browsing branches using Launchpad

#### Finding available branches

While there are many advantages in adopting distributed version control, one of the things that disappears is the all-knowing central server with knowledge about all available branches. Indeed in a distributed environment, interesting branches can literally exist in 100s of locations across the Internet (or within an Intranet for that matter).

Launchpad fills this gap by providing a registry of branches.

#### Registering branches

Branches can be uploaded to Launchpad or simply registered as being available in an external location. Branches can also be given a Status such as *New*, *Development*, *Mature* or *Abandoned*.

Note: External branches can even be hosted in legacy version control tools, i.e. CVS and Subversion. Code in these systems will be scanned and converted to Bazaar branches on a periodic basis. For maximum fidelity of course, it is preferable for external branches to be hosted in Bazaar.

#### **Browsing branches**

Branches can be listed, filtered and sorted by numerous attributes including Name, Registrant, Author, Status, Age and time of last commit. Browsing of branches is also provided making it easy to see things such as:

- where the branch can be downloaded from
- how to upload changes
- recent commits and the changes made by each
- the source code of individual files for a given version.

# Accessing code in Launchpad using Bazaar

#### Getting the code for an open source project

As Launchpad keeps track of thousands of open source projects and their latest code whether it be managed by Bazaar, CVS or Subversion, Bazaar users can grab that code as easily as this:

```
bzr branch lp:project-name
```

where *project-name* is the Launchpad project ID. Here are some examples:

```
bzr branch lp:inkscape
bzr branch lp:amarok
bzr branch lp:python
bzr branch lp:rails
bzr branch lp:java-gnome
```

You can then browse the code locally using your favorite editor or IDE and change the code if you wish.

If a project has multiple series registered (e.g. a development series and a maintenance series), the latest code for a given series can be fetched using:

```
bzr branch lp:project-name/series
```

## Publishing your changes

Having fixed that annoying bug or added that cool feature you've always wanted, it's time to impress your friends and make the world a better place by making your code available to others. As explained earlier, Launchpad is a free Bazaar code hosting service so you can

push your branch to it and others can access your code from there. For example, assuming you are a member of the relevant team, login to launchpad like this:

```
bzr launchpad-login userid
```

where *userid* is your Launchpad user ID. You can then push your changes to a team branch like this:

```
bzr push lp:~team-name/project-name/branch-name
```

Others can then download your code like this:

```
bzr branch lp:~team-name/project-name/branch-name
```

#### Personal branches

Even if you are not a member of a team, Launchpad can be used to publish your changes. In this case, simply create a personal branch like this:

```
bzr push lp:~userid/project-name/branch-name
```

Others can then download your code like this:

```
bzr branch lp:~userid/project-name/branch-name
```

Note: Even when publishing to a personal branch, it is polite to notify the upstream developers about your branch so they can pull your changes from it if they are generally applicable to all users and meet the project's quality standards.

# Linking branches using Launchpad

#### Associating a branch with a bug

After registering a branch, you can associate it to a bug so that people interested in that bug can track and download the fix as it becomes available.

To do this, the steps are:

- 1. Navigate to the bug in question.
- 2. Select Add branch under Actions.
- 3. Select the branch.
- 4. Optionally set the State of the relationship. This is *Fix In Progress* by default but you may wish to set it to another state such as *Fix Available* if the branch already addresses the issue.

If you wish, you can also provide some arbitrary comments about the relationship between the bug and the branch.

# Changing the state in Launchpad while committing in Bazaar

Bazaar and Launchpad can work together to reduce some of the status housekeeping for you. When you commit using Bazaar, use the –fixes option like this:

```
bzr commit --fixes lp:1234 -m "..."
```

where 1234 is the bug ID. This will changes the State of the bugbranch relationship to *Fix Available*. If the one commit fixes multiple issues, the –fixes option can be specified multiple times.

One of the cool things about this feature is that Launchpad does not need to be accessible when making the commit. The --fixes option works by storing metadata which Launchpad will detect next time the branch is pushed to it or scanned once online again.

Note: Launchpad will not implicitly close a bug just because a branch is available that fixes it. There are several reasons for this. Firstly, the branch usually needs to be merged into the trunk (main development branch) before most teams consider it fixed. Secondly, many teams have a separate process for confirming bugs are fixed over and above a developer saying so.

As explained later, merge control features are currently under development in Launchpad and automatically changing the status of bugs to *Fix Committed* will be more appropriate once those features are in place.

#### Associating a branch with a blueprint

After registering a branch, you can associate it to a blueprint so that people interested in that blueprint can track and test the feature as it develops.

To do this, the steps are:

- 1. Navigate to the blueprint in question.
- 2. Select Link branch under Actions.
- 3. Select the branch.

If you wish, you can also provide some arbitrary comments about the relationship between the blueprint and the branch.

# Managing releases using Launchpad

#### Integrating changes

Once a branch has been developed and published, communities typically go through a rigorous process before those changes are integrated into the core product and rolled out to end users. Some of the steps involved may include:

- peer review of the changes
- deciding which releases to include the changes in, e.g. the next maintenance release, the next major release, or both
- running functional regression tests
- benchmarking to ensure performance remains acceptable
- packaging into early access releases for end user testing
- documentation updates, e.g. Release Notes for the targeted releases
- translation of the user interface and documentation into multiple languages.

This section briefly looks at some of the features in Launchpad that help get good quality code into production. Strong integration with Bazaar is core to making this happen smoothly.

Note: Where indicated, some of the features below are still under development. If one or more of these features interest you, please consider joining the Launchpad beta test team at this link: <a href="https://help.launchpad.net/JoiningLaunchpadBetaTesters">https://help.launchpad.net/JoiningLaunchpadBetaTesters</a>. You can then get early access to features and provide feedback to the developers before wider roll-out.

### Branch merge proposals

After navigating to a branch in Launchpad, one of the available actions is *Propose for merging*. This lets you nominate which branch this code ought to be merged into.

Tracking the knowledge about which branches are proposed to be merged into a codeline helps Release Managers keep on top of what still needs to be completed, or can be completed, before a ship date. Using this information, they can ensure branches are merged after completing any necessary reviews. In the simple case, the Release Manager may manually merge branches. In more advanced cases, the merging could be automatically done by a robot (like PQM) when the branch reaches the right state (e.g. *Review completed*).

#### Code review tracking

A number of features are under development in Launchpad to track the states, conversations and outcomes of code reviews. These features are expected to be integrated with branch merge proposals and branch browsing features.

# Personal Package Archives (PPAs)

PPAs help developers and development teams get custom builds into the hands of users for early testing and feedback. In other words, a PPA allows a developer to form a community of testers who are interested in their changes. The testing community can install the packages, run them for the test period and then remove them cleanly from their system.

See https://help.launchpad.net/PPAQuickStart for further details.

#### **Translations**

The Translations module in Launchpad is designed to make it easy

for anyone to get involved translating applications to languages they know. Translators are shielded from the low level details.

Launchpad keeps track of the translations for each major version of a project separately, allowing translators to continue to improve the translations of your stable releases while others start work on newer versions that are still in development. Translation speed in reduced by sharing resources across projects. Automatic suggestions, from a library of 750,000 translated strings, and a community of 19,000 registered translators can radically cut the time required to localise your project into many languages.

## Summary

The communities we join, whether off-line or on-line, say a lot about the sort of people we are. The flip-side to this is that the tools you choose for your community - particularly the CDE and version control tool - can have a large impact on who joins and how easily they can contribute.

In their own right, Launchpad and Bazaar are highly useful tools. Together, they can:

- help your community track major assets such as source code and knowledge
- help it grow by reducing barriers to entry
- help it interact with related communities.

In particular, Launchpad is a free code hosting service for your Bazaar branches, branches can be browsed online, branches can be linked to bugs and blueprints, and the status of bug-branch relationships can be automatically managed by mentioning the bug while committing in Bazaar. Further integration is under development with the aim of streamlining the process from *great idea* to *running code in the hands of end users*.

If you have any feedback on how you'd like to see Bazaar and Launchpad further integrated, please contact us on the Bazaar mailing list, bazaar@lists.canonical.com.

While designed as a free service to support open source projects, Canonical may make Launchpad available to commercial software developers depending on their requirements. We would be happy to hear from you if you think Launchpad would be useful for managing your community, open source or otherwise.

# **Centralized Workflow Tutorial**

#### Overview

This document describes a possible workflow for using Bazaar. That of using Bazaar, the distributed version control system, in a centralized manner. Bazaar is designed to be very flexible and allows several different workflows, from fully decentralized to mostly centralized. The workflow used here is meant to ease a new user into more advanced usage of Bazaar, and allow them to work in a mix of centralized and decentralized operations.

In general, this document is meant for users coming from a background of centralized version control systems such as CVS or subversion. It is common in work settings to have a single central server hosting the codebase, with several people working on this codebase, keeping their work in sync. This workflow is also applicable to a single developer working on several different machines.

### **Initial Setup**

These are some reasonably simple steps to setup Bazaar so that it works well for you.

#### Setting User Email

Your user identity is stored with each commit. While this doesn't have to be accurate or unique, it will be used in log messages and annotations, so it is better to have something real.

```
% bzr whoami "John Doe <jdoe@organization.com>"
```

### Setting up a local Repository

Bazaar branches generally copy the history information around with them, which is part of how you can work in a fully decentralized manner. As an optimization, it is possible for related branches to combine their storage needs so that you do not need to copy around all of this history information whenever you create a new branch.

The best way to do this is to create a Shared Repository. In general, branches will share their storage if they exist in a subdirectory of a Shared Repository. So let's set up a Shared Repository in our home directory, thus all branches we create underneath will share their history storage.

```
% bzr init-repo --trees ~
```

#### Setting up a remote Repository

Many times you want a location where data is stored separately from where you do your work. This workflow is required by centralized

systems (CVS/SVN). Usually they are on separate machines, but not always. This is actually a pretty good setup, especially in a work environment. Since it ensures a central location where data can be backed up, and means that if something happens to a developer's machine, no committed work has to be lost.

So let's set up a shared location for our project on a remote machine called centralhost. Again, we will use a Shared Repository to optimize disk usage.

% bzr init-repo --no-trees sftp://centralhost/srv/bzr/

You can think of this step as similar to setting up a new cvsroot, or subversion repository. The --no-trees option tells bzr to not populate the directory with a working tree. This is appropriate, since no one will be making changes directly in the branches within the central repository.

# Migrating an existing project to Bazaar

Now that we have a repository, let's create a versioned project. Most of the time, you will already have some code that you are working with, that you now want to version using Bazaar. If the code was originally in source control, there are many ways to convert the project to Bazaar without losing any history. However, this is outside the scope of this document. See Tracking Upstream for some possibilities (section "Converting and keeping history").

### Developer 1: Creating the first revision

So first, we want to create a branch in our remote Repository, where we want to host the project. Let's assume we have a project named "sigil" that we want to put under version control.

```
% bzr init sftp://centralhost/srv/bzr/sigil
```

This can be thought of as the "HEAD" branch in CVS terms, or as the "trunk" in Subversion terms. We will call this the dev branch.

I prefer working in a subdirectory of my home directory to avoid collisions with all the other files that end up there. Also, we will want a project directory where we can hold all of the different branches we end up working on.

```
% cd ~
% mkdir work
% cd work
% mkdir sigil
% cd sigil
% bzr checkout sftp://centralhost/srv/bzr/sigil dev
% cd dev
% cd dev
% cp -ar ~/sigil/* .
% bzr add
% bzr commit -m "Initial import of Sigil"
```

In the previous section, we created an empty branch (the /sigil branch) on centralhost, and then checkout out this empty branch onto our workstation to add files from our existing project. There are many ways to set up your working directory, but the steps above make it easy to handle working with feature/bugfix branches. And one of the strong points of Bazaar is how well it works with branches.

At this point, because you have a 'checkout' of the remote branch, any commits you make in ~/work/sigil/dev/ will automatically be saved both locally, and on centralhost.

#### Developer N: Getting a working copy of the project

Since the first developer did all of the work of creating the project, all other developers would just checkout that branch. **They should still follow** Setting User Email and Setting up a local Repository.

To get a copy of the current development tree:

```
% cd ~/work/sigil
% bzr checkout sftp://centralhost/srv/bzr/sigil dev
```

Now that two people both have a checkout of sftp://centralhost/srv/bzr/sigil, there will be times when one of the checkouts will be out of date with the current version. At commit time, Bazaar will inform the user of this and prevent them from committing. To get up to date, use bzr update to update the tree with the remote changes. This may require resolving conflicts if the same files have been modified.

# Developing on separate branches

So far everyone is working and committing their changes into the same branch. This means that everyone needs to update fairly regularly and deal with other people's changes. Also, if one person commits something that breaks the codebase, then upon syncing, everyone will get the problem.

Usually, it is better to do development on different branches, and then integrate those back into the main branch, once they are stable. This is one of the biggest changes from working with CVS/SVN. They both allow you to work on separate branches, but their merging algorithms are fairly weak, so it is difficult to keep things synchronized. Bazaar tracks what has already been merged, and can even apply changes to files that have been renamed.

#### Creating and working on a new branch

We want to keep our changes available for other people, even if they aren't quite complete yet. So we will create a new public branch on centralhost, and track it locally.

We now have a place to make any fixes we need to doodle. And we would not interrupt people who are working on other parts of the code. Because we have a checkout, any commits made in the ~/work/sigil/doodle-fixes/ will also show up on centralhost. [1] It is also possible to have two developers collaborate on one of these

branches, just like they would have collaborated on the dev branch. [2]

- It may look odd to have a branch in a subdirectory of another branch. This is just fine, and you can think of it as a hierarchical namespace where the nested branch is derived from the outer branch.
  - (1, 2) When using lots of independent branches, having to retype the full URL all the time takes a lot of typing. We are looking into various methods to help with this, such as branch aliases, etc. For now, though, the bzrtools plugin provides the
- [2] bzr cbranch command. Which is designed to take a base branch, create a new public branch, and create a checkout of that branch, all with much less typing. Configuring cbranch is outside the scope of this document, but the final commands are similar to:

```
% bzr cbranch dev my-feature-branch
```

### Merging changes back

When it is decided that some of the changes in doodle-fixes are ready to be merged into the main branch, simply do:

```
% cd ~/work/sigil/dev
% bzr merge ../doodle-fixes
```

Now the changes are available in the dev branch, but they have not been committed yet. This is the time when you want to review the final changes, and double check the code to make sure it compiles cleanly and passes the test suite. The commands bzr status and bzr diff are good tools to use here. Also, this is the time to resolve any conflicts. Bazaar will prevent you from committing until you have resolved these conflicts. That way you don't accidentally commit the conflict markers. The command bzr status will show the conflicts along with the other changes, or you can use bzr conflicts to just

list conflicts. Use bzr resolve file/name or bzr resolve --all once conflicts have been handled. [3] If you have a conflict that is particularly difficult to solve you may want to use the bzr remerge command. It will let you try different merge algorithms, as well as let you see the original source lines (--show-base).

Some systems make you resolve conflicts as part of the merge process. We have found that it is usually easier to resolve

[3] conflicts when you have the view of the entire tree, rather than

just a single file. It gives you much more context, and also lets you run tests as you resolve the problems.

### **Recommended Branching**

One very common way to handle all of these branches is to give each developer their own branch, and their own place to work in the central location. This can be done with:

This gives each developer their own branch to work on. And, they can easily create a new feature branch for themselves with just [2]

## Glossary

### **Shared Repository**

Bazaar has the concept of a "Shared Repository". This is similar to the traditional concept of a repository in other VCSs like CVS and Subversion. For example, in Subversion you have a remote repository, which is where all of the history is stored, and locally you don't have any history information, only a checkout of the working tree files. Note that "Shared" in this context means shared between branches. It *may* be shared between people, but standalone branches can also be shared between people.

In Bazaar terms, a "Shared Repository" is a location where multiple branches can **share** their revision history information. In order to support decentralized workflows, it is possible for every branch to store its own revision history information. But this is often inefficient, since related branches share history, and they might as well share the storage as well.

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

### bzr 2.2b1

**2.2b1**: 2010-04-01

This is the first beta of the 2.2 series, leading up to a 2.2.0 release in July or August. Beta releases are suitable for everyday use but may cause some incompatibilities with plugins. Some plugins may need small updates to work with 2.2b1.

2.2b1 includes some changes to make merge conflicts easier to understand and resolve. It also removes some old unnecessary code, and loads somewhat less code at startup. It starts adding a common infrastructure for dealing with colocated named branches, which can be implemented in various ways in either bzr native or foreign formats. On Ubuntu and other platforms with the apport bugreporting library, there's an easier path to report problems with bzr. We plan to continue with these themes through the 2.2 series.

Over thirty bugs have been fixed, including in the log command, exporting to tarballs, restarting interrupted system calls, portability of compiled extensions, making backups during upgrade, and locking on ftp.

## **Compatibility Breaks**

- BTreeGraphIndex can now take an offset to indicate that the data starts somewhere other than then beginning of the file. (John Arbash Meinel)
- Deleted very old hidden commands versionedfile-list, weave-plan-merge, weave-merge-text. (Martin Pool)
- Repository.get\_inventory\_sha1() and Repository.get\_revision\_xml() have been removed. (Jelmer Vernooij)
- Repository.get\_revision\_inventory() has been removed in favor of Repository.get\_inventory(). (Jelmer Vernooij)
- All test servers have been moved out of the bzrlib.transport hierarchy to bzrlib.tests.test\_server except for MemoryServer, ChrootServer and PathFilteringServer. bzrlib users may encounter test failures that can be fixed by updating the related imports from bzrlib.transport.xxx to bzrlib.tests.test\_server. (Vincent Ladeuil)
- BranchReferenceFormat.initialize() now takes an optional name argument as its second parameter, for consistency with the initialize() method of other formats. (Jelmer Vernooij)

#### **New Features**

- Added bzr remove-branch command that can remove a local or remote branch. (Jelmer Vernooij, #276295)
- bzr export now takes an optional argument --per-filetimestamps to set file mtimes to the last timestamp of the last revision in which they were changed rather than the current time. (Jelmer Vernooij)
- If the Apport crash-reporting tool is available, bzr crashes are now stored into the /var/crash apport spool directory, and the user is invited to report them to the developers from there, either automatically or by running apport-bug. No information is sent without specific permission from the user. (Martin Pool, #515052)
- Parsing of command lines, for example in diff --using, no longer treats backslash as an escape character on Windows. (Gordon Tyler, #392248)
- Plugins can be disabled by defining BZR\_DISABLE\_PLUGINS as a list of plugin names separated by ':' (';' on windows). (Vincent Ladeuil, #411413)
- Plugins can be loaded from arbitrary locations by defining BZR\_PLUGINS\_AT as a list of name@path separated by ":" (";" on windows). This takes precedence over BZR\_PLUGIN\_PATH for the specified plugins. This is targeted at plugin developers for punctual needs and *not* intended to replace BZR\_PLUGIN\_PATH. (Vincent Ladeuil, #82693)
- Tag names can now be determined automatically by automatic\_tag\_name hooks on Branch if they are not specified on the command line. (Jelmer Vernooij)
- Tree-shape conflicts can be resolved by providing --take-this and --take-other to the bzr resolve command. Just marking

- the conflict as resolved is still accessible via the --done default action. (Vincent Ladeuil)
- Merges can be proposed on Launchpad with the new lppropose-merge command. (Aaron Bentley, Jonathan Lange)

### **Bug Fixes**

- Added docstring for Tree.iter\_changes (John Arbash Meinel, #304182)
- Allow additional arguments to RemoteRepository.add\_inventory\_by\_delta(). (Jelmer Vernooij, #532631)
- Allow exporting a single file using bzr export. (Michal Junák, #511987)
- Allow syscalls to automatically restart when TextUIFactory's SIGWINCH handler is invoked, avoiding EINTR errors during blocking IO, which are often poorly handled by Python's libraries and parts of bzrlib. (Andrew Bennetts, #496813)
- Avoid infinite recursion when probing for apport. (Vincent Ladeuil, #516934)
- Avoid malloc(0) in patiencediff, which is non-portable. (Martin Pool, #331095)
- Avoid truncating svn URLs. (Martin Pool, Martin von Gagern, #545185)
- bzr add will not add conflict related files unless explicitly required. (Vincent Ladeuil, #322767, #414589)
- bzr dump-btree now works on \*.cix and \*.six files. Those indices do not have reference lists, so dump-btree will simply show None instead. (Andrew Bennetts, #488607)
- bzr help will no longer trigger the get\_missing\_command hook

when doing a topic lookup. This avoids prompting (like 'no command plugins/loom, did you mean log?') when getting help. In future we may trigger the hook deliberately when no help topics match from any help index. (Robert Collins, #396261)

- bzr log -no -r..A.B.c should not crash but just consider the None revspec as representing the first revision of the branch. (Vincent Ladeuil, #519862)
- bzr pull now uses the correct URL when a Launchpad URI is explicitly defined as the pull location in locations.conf. Strings like lp:bzr can now be correctly joined to other URLs. (Gordon Tyler, #534787)
- bzr remove-tree can now remove multiple working trees. (Jared Hance, Andrew Bennetts, #253137)
- bzr resolve --take-this and --take-other now correctly renames the kept file on content conflicts where one side deleted the file. (Vincent Ladeuil, #529968)
- bzr upgrade now creates the backup.bzr directory with the same permissions as .bzr directory on a POSIX OS. (Parth Malwankar, #262450)
- bzr upgrade now names backup directory as backup.bzr.~N~ instead of backup.bzr. This directory is ignored by bzr commands such as add. (Parth Malwankar, #335033, #300001)
- Correctly interpret "451 Rename/move failure: Directory not empty" from ftp servers while trying to take a lock. (Martin Pool, #528722)
- DirStateRevisionTree.kind() was returning wrong result when 'kind' changes occured between the workingtree and one of its

parents. (Vincent Ladeuil, #535547)

- Fix log to better check ancestors even if merged revisions are involved. (Vincent Ladeuil, #476293)
- Many IO operations that returned EINTR were retried even if it wasn't safe to do so via careless use of until\_no\_eintr. Bazaar now only retries operations that are safe to retry, and in some cases has switched to operations that can be retried (e.g. sock.send rather than sock.sendall). (Andrew Bennetts, Martin <gzlist@googlemail.com>, #496813)
- Path conflicts now support –take-this and –take-other even when a deletion is involved. (Vincent Ladeuil, #531967)
- Network transfer amounts and rates are now displayed in SI units according to the Ubuntu Units Policy <a href="https://wiki.ubuntu.com/UnitsPolicy">https://wiki.ubuntu.com/UnitsPolicy</a>. (Gordon Tyler, #514399)
- Support kind markers for socket and fifo filesystem objects. This
  prevents bzr status --short from crashing when those files are
  present. (John Arbash Meinel, #303275)
- bzr mkdir DIR will not create DIR unless DIR's parent is a versioned

directory. (Parth Malwankar, #138600)

- SSH child processes will now ignore SIGQUIT on nix systems so breaking into the debugger won't kill the session. (Martin <gzlist@googlemail.com>, #162502)
- Tolerate patches with leading noise in <a href="mailto:bzr-handle-patch">bzr-handle-patch</a>. (Toshio Kuratomi, Martin Pool, #502076)
- .bazaar, .bazaar/bazaar.conf and .bzr.log inherit user and

group ownership from the containing directory. This allow bzr to work better with sudo. (Parth Malwankar, #376388)

### **API Changes**

- bzrlib.merge\_directive.\_BaseMergeDirective has been renamed to bzrlib.merge\_directive.BaseMergeDirective and is now public. (Jelmer Vernooij)
- BranchFormat.initialize now takes an optional name of the colocated branch to create. (Jelmer Vernooij)
- BzrDir.get\_branch\_transport now takes an optional name of the colocated branch to open. (Jelmer Vernooij)
- Added bzrlib.osutils.set\_signal\_handler, a convenience function that can set a signal handler and call signal.siginterrupt(signum, False) for it, if the platform and Python version supports it. (Andrew Bennetts, #496813)
- New bzrlib.initialize is recommended for programs using bzrlib to run when starting up; it sets up several things that previously needed to be done separately. (Martin Pool, #507710)
- Exporters now support a per\_file\_timestamps argument to write out the timestamp of the commit in which a file revision was introduced. (Jelmer Vernooij)
- New method BzrDir.list\_branches() that returns a sequence of branches present in a control directory. (Jelmer Vernooij)
- New transport methods readlink, symlink and hardlink. (Neil Santos)
- Remove unused commandFailed exception. (Martin Pool)

#### **Internals**

- bzrlib.branchbuilder.BranchBuilder.build\_snapshot
   accepts a message\_callback in the same way that commit does.
   (Robert Collins)
- bzrlib.builtins.Commit.run
   raises
   bzrlib.errors.BoundBranchOutOfDate
   rather
   than
   bzrlib.errors.BzrCommandError when the bound branch is out
   of date. (Gary van der Merwe)
- bzrlib.commands.run\_bzr is more extensible: callers can supply the functions to load or disable plugins if they wish to use a different plugin mechanism; the —help, —version and nocommand name code paths now use the generic pluggable command lookup infrastructure. (Robert Collins)
- bzrlib.errors.BoundBranchOutOfDate has a new field extra\_help which can be set to add extra help to the error. (Gary van der Merwe)
- New method Branch.automatic\_tag\_name that can be used to find the tag name for a particular revision automatically. (Jelmer Vernooij)
- The methods BzrDir.create\_branch(), BzrDir.destroy\_branch() and BzrDir.open\_branch() now take an optional name argument. (Jelmer Vernooij)

# **Testing**

- bzr now has a .testr.conf file in its source tree configured appropriately for running tests with Testrepository (https://launchpad.net/testrepository). (Robert Collins)
- Documentation about testing with subunit has been tweaked.
   (Robert Collins)
- Known failures has been added for resolve –take-other on ParentLoop conflicts. This reflects bug #537956 without fixing it. (Vincent Ladeuil)
- New <a href="bzrlib.tests.test\_import\_tariff">bzrlib.tests.test\_import\_tariff</a> can make assertions about what Python modules are loaded, to guard against startup time or library dependency regressions. (Martin Pool)
- Stop sending apport crash files to .cache in the directory from which bzr selftest was run. (Martin Pool, #422350)
- Tests no longer fail if "close() called during concurrent operation on the same file object" occurs when closing the log file (which can happen if a thread tries to write to the log file at the wrong moment). An warning will be written to stderr when this happens, and another warning will be written if the log file could not be closed after retrying 100 times. (Andrew Bennetts, #531746)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

# bzr 2.1.1

**2.1.1:** 2010-03-24

This is a small bugfix release. Upgrading is recommended for anyone running 2.1.0 or earlier.

### **Bug Fixes**

- Allow syscalls to automatically restart when TextulFactory's SIGWINCH handler is invoked, avoiding EINTR errors during blocking IO, which are often poorly handled by Python's libraries and parts of bzrlib. (Andrew Bennetts, #496813)
- Avoid malloc(0) in patiencediff, which is non-portable. (Martin Pool, #331095)
- Fix plugin packaging on Windows. (Ian Clatworthy, #524162)
- Fix stub sftp test server to call os.getcwdu(). (Vincent Ladeuil, #526221, #526353)
- Fixed CHM generation by moving the NEWS section template into a separate file. (lan Clatworthy, #524184)
- Merge correctly when this\_tree is not a WorkingTree. (Aaron Bentley)
- Register SIGWINCH handler only when creating a TextUIFactory; avoids problems importing bzrlib from a non-main thread. (Elliot Murphy, #521989)
- Standardize the error handling when creating a new StaticTuple (problems will raise TypeError). (Matt Nordhoff, #457979)
- Warn if pyrex is too old to compile the new <u>simpleset</u> and <u>staticTuple</u> extensions, rather than having the build fail randomly. (John Arbash Meinel, #449776)

# Documentation

- Added a link to the Desktop Guide. (Ian Clatworthy)
- Added What's New in Bazaar 2.1 document. (Ian Clatworthy)
- Drop Google Analytics from the core docs as they caused problems in the CHM files. (Ian Clatworthy, #502010)

# **API Changes**

 Added bzrlib.osutils.set\_signal\_handler, a convenience function that can set a signal handler and call signal.siginterrupt(signum, False) for it, if the platform and Python version supports it. (Andrew Bennetts, #496813)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

# bzr 2.1.0

**Codename:** Strasbourg **2.1.0:** 2010-02-11

This release marks our second long-term-stable series. The Bazaar team has decided that we will continue to make bugfix-only 2.0.x and 2.1.x releases, along with 2.2 development releases.

This is a fairly incremental update, focusing on polish and bugfixing. There are no changes for supported disk formats. Key updates include reduced memory consumption for many operations, a new per-file merge hook, ignore patterns can now include '!' to exclude files, globbing support for all commands on Windows, and support for addressing home directories via bzr+ssh://host/~/ syntax.

Users are encouraged to upgrade from the 2.0 stable series.

### **Bug Fixes**

- Don't require testtools to use sftp. (Vincent Ladeuil, #516183)
- Fix "AttributeError in Inter1and2Helper" during fetch. (Martin Pool, #513432)
- bzr update performs the two merges in a more logical order and will stop when it encounters conflicts. (Gerard Krol, #113809)
- Give a better error message when doing bzr bind in an already bound branch. (Neil Martinsen-Burrell, #513063)
- Ignore KeyError from remove\_index during \_abort\_write\_group in a pack repository, which can happen harmlessly if the abort occurs during finishing the write group. Also use bzrlib.cleanup so that any other errors that occur while aborting the individual packs won't be hidden by secondary failures when removing the corresponding indices. (Andrew Bennetts, #423015)
- Set the mtime of files exported to a directory by bzr export all to the same value to avoid confusing make and other date-based build systems. (Robert Collins, #515631)

# **Improvements**

• Fetching into experimental formats will now print a warning. (Jelmer Vernooij)

# **API Changes**

- Repository.deserialise\_inventory has been renamed to Repository.\_deserialise\_inventory to indicate it is private. (Jelmer Vernooij)
- Repository.get\_inventory\_xml has been renamed to Repository.get\_inventory\_xml to indicate it is private. (Jelmer Vernooij)
- Repository.serialise\_inventory has been renamed to Repository.\_serialise\_inventory to indicate it is private.
- Using the bzrlib.chk\_map module from within multiple threads at the same time was broken due to race conditions with a module level page cache. This shows up as a KeyError in the bzrlib.lru\_cache code with bzrlib.chk\_map in the backtrace, and can be triggered without using the same high level objects such as bzrlib.repository.Repository from different threads. chk\_map now uses a thread local cache which may increase memory pressure on processes using threads. (Robert Collins, John Arbash Meinel, #514090)
- The new merge\_file\_content should now be ok with tests to avoid regressions. (Vincent Ladeuil, #515597)

# **Internals**

• Use bzrlib.cleanup rather than less robust try/finally blocks in several places in bzrlib.merge. This avoids masking prior errors when errors like ImmortalPendingDeletion occur during cleanup in do\_merge. (Andrew Bennetts, #517275)

# **API Changes**

The remove\_index method of bzrlib.repofmt.pack\_repo.AggregateIndex no longer takes a pack argument. This argument was always ignored. (Andrew Bennetts, #423015)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

# bzr 2.1.0rc2

Codename: after the bubbles

**2.1.0rc2**: 2010-01-29

This is a quick-turn-around to update a small issue with our new perfile merge hook. We expect no major changes from this to the final 2.1.0.

### **API Changes**

• The new merge\_file\_content hook point has been altered to provide a better API where state for extensions can be stored rather than the too-simple function based approach. This fixes a performance regression where branch configuration would be parsed per-file during merge. As part of this the included news\_merger has been refactored into a base helper class bzrlib.merge.ConfigurableFileMerger. (Robert Collins, John Arbash Meinel, #513822)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release

Notes »

previous | next

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

#### bzr 2.1.0rc1

Codename: the 'new' stable

**2.1.0rc1**: 2009-01-21

This is the first stable release candidate for Bazaar's 2.1 series. From this point onwards, the 2.1 series will be considered stable (as the 2.0 series) and only bugfixes are expected to be incorporated. The dozen or so bugfixes in the 2.0.4 release are also included in this release (along with more than 15 more bugfixes). Some of the interesting features are support for per-file merge hooks, bzr unshelve --preview, support for using! in ignore files to exclude files from being ignored, a small memory leak was squashed, and many ObjectNotLocked errors were fixed. This looks to be a very good start for a new stable series.

#### **New Features**

- Add bug information to log output when available. (Neil Martinsen-Burrell, Guillermo Gonzalez, #251729)
- Added merge\_file\_content hook point to Merger, allowing plugins to register custom merge logic, e.g. to provide smarter merging for particular files.
- Bazaar now includes the news\_merge plugin. It is disabled by default, to enable it add a news\_merge\_files option to your configuration. Consult bzr help news\_merge for more information. (Andrew Bennetts)
- bzr branch now takes a --bind option. This lets you branch and bind all in one command. (Ian Clatworthy)
- bzr switch now takes a --revision option, to allow switching to a specific revision of a branch. (Daniel Watkins, #183559)
- bzr unshelve --preview can now be used to show how a patch on the shelf would be applied to the working tree. (Guilherme Salgado, #308122)
- bzr update now takes a --revision argument. This lets you change the revision of the working tree to any revision in the ancestry of the current or master branch. (Matthieu Moy, Mark Hammond, Martin Pool, #45719)
- -Dbytes can now be used to display the total number of bytes transferred for the current command. This information is always logged to .bzr.log for later inspection. (John Arbash Meinel)
- New ignore patterns. Patterns prefixed with '!' are exceptions to ignore patterns and take precedence over regular ignores. Such exceptions are used to specify files that should be versioned which would otherwise be ignored. Patterns prefixed with '!!' act as regular ignore patterns, but have highest precedence, even over the '!' exception patterns. (John Whitley, #428031)
- The supress\_warnings configuration option has been introduced

to disable various warnings (it currently only supports the format\_deprecation warning). The new option can be set in any of the following locations: bazaar.conf, locations.conf and/or branch.conf. (Ted Gould, Matthew Fuller, Vincent Ladeuil)

#### **Bug Fixes**

- Always show a message if an OS error occurs while trying to run a user-specified commit message editor. (Martin Pool, #504842)
- bzr diff will now use the epoch when it is unable to determine the timestamp of a file, if the revision it was introduced in is a ghost. (Jelmer Vernooij, #295611)
- bzr switch -b can now create branches that are located using directory services such as 1p:, even when the branch name doesn't contain a '/'. (Neil Martinsen-Burrell, #495263)
- bzr unshelve has improved messages about what it is doing.
   (Neil Martinsen-Burrell, #496917)
- Concurrent autopacking is more resilient to already-renamed pack files. If we find that a file we are about to obsolete is already obsoleted, we do not try to rename it, and we leave the file in obsolete\_packs. The code is also fault tolerant if a file goes missing, assuming that another process already removed the file. (John Arbash Meinel, Gareth White, #507557)
- Fix "Too many concurrent requests" in reconcile when network connection fails. (Andrew Bennetts, #503878)
- Fixed a side effect mutation of RemoteBzrDirFormat.\_network\_name that caused some tests to fail when run in a non-default order. Probably no user impact. (Martin Pool, #504102)
- Fixed ObjectNotLocked error in bzr cat -rbranch:../foo FILE. (Andrew Bennetts, #506274)
- FTP transports support Unicode paths by encoding/decoding them as utf8. (Vincent Ladeuil, #472161)
- Listen to the SIGWINCH signal to update the terminal width. (Vincent Ladeuil, #316357)
- Progress bars are now hidden when --quiet is given. (Martin

Pool, #320035)

- SilentUIFactory now supports make\_output\_stream and discards whatever is written to it. This un-breaks some plugin tests that depended on this behaviour. (Martin Pool, #499757)
- When operations update the working tree, all affected files should end up with the same mtime. (eg. when versioning a generated file, if you update the source and the generated file together, the generated file should appear up-to-date.) (John Arbash Meinel, Martin <gzlist>, #488724)

#### **Improvements**

- Added add\_cleanup and cleanup\_now to bzrlib.command.Command.
   All the builtin commands now use add\_cleanup rather than try/finally blocks where applicable as it is simpler and more robust. (Andrew Bennetts)
- All except a small number of storage formats are now hidden, making the help for numerous commands far more digestible. (lan Clatworthy)
- Attempts to open a shared repository as a branch (e.g. bzr branch path/to/repo) will now include "location is a repository" as a hint in the error message. (Brian de Alwis, Andrew Bennetts, #440952)
- Push will now inform the user when they are trying to push to a foreign VCS for which roundtripping is not supported, and will suggest them to use dpush. (Jelmer Vernooij)
- The version of bzr being run is now written to the log file.
   (\_\_monty\_\_\_, #257170)
- Transport network activity indicator is shown more of the time when Bazaar is doing network IO. (Martin Pool)

#### **Documentation**

- Add documentation on creating merges with more than one parent. (Neil Martinsen-Burrell, #481526)
- Better explain the –uncommitted option of merge. (Neil Martinsen-Burrell, #505088)
- Improve discussion of pending merges in the documentation for revert. (Neil Martinsen-Burrell, #505093)
- Improved help for bzr send. (Martin Pool, Bojan Nikolic)
- There is a System Administrator's Guide in <a href="doc/en/admin-guide">doc/en/admin-guide</a>, including discussions of installation, relevant plugins, security and backup. (Neil Martinsen-Burrell)
- The conflicts help topic has been renamed to conflict-types. (lan Clatworthy)
- The User Reference is now presented as a series of topics.
   Many of the included topics have link and format tweaks applied. (lan Clatworthy)

### **API Changes**

- Added cachedproperty decorator to bzrlib.decorators. (Andrew Bennetts)
- Many test features were renamed from FooFeature to foo\_feature to be consistent with instances being lower case and classes being CamelCase. For the features that were more likely to be used, we added a deprecation thunk, but not all. (John Arbash Meinel)
- Merger classes (such as Merge3Merger) now expect a this\_branch parameter in their constructors, and provide this\_branch as an attribute. (Andrew Bennetts)
- The Branch hooks pre\_change\_branch\_tip no longer masks exceptions raised by plugins - the original exceptions are now preserved. (Robert Collins)
- The Transport server.tearDown method is now renamed to stop\_server and setUp to start\_server for consistency with our normal naming pattern, and to avoid confusion with Python's TestCase.tearDown. (Martin Pool)
- WorkingTree.update implementations must now accept a revision parameter.

#### **Internals**

- Added BzrDir.open\_branchv3 smart server request, which can receive a string of details (such as "location is a repository") as part of a nobranch response. (Andrew Bennetts, #440952)
- New helper osutils.UnicodeOrBytesToBytesWriter which encodes unicode objects but passes str objects straight through. This is used for selftest but may be useful for diff and other operations that generate mixed output. (Robert Collins)
- New exception NoRoundtrippingSupport, for use by foreign branch plugins. (Jelmer Vernooij)

#### **Testing**

- bzrlib.tests.permute\_for\_extension is a helper that simplifies running all tests in the current module, once against a pure python implementation, and once against an extension (pyrex/C) implementation. It can be used to dramatically simplify the implementation of load\_tests. (John Arbash Meinel)
- bzrlib.tests.TestCase
   now
   subclasses
   testtools.testcase.TestCase
   This permits features in testtools
   such as getUniqueInteger and getUniqueString to be used
   Because of this, testtools version 0.9.2 or newer is now a dependency to run bzr selftest
   Running with versions of testtools less than 0.9.2 will cause bzr to error while loading the test suite
   (Robert Collins)
- Shell-like tests now support the command "mv" for moving files. The syntax for mv file1 file2, mv dir1 dir2 and mv file dir is supported. (Neil Martinsen-Burrell)
- The test progress bar no longer distinguishes tests that 'errored' from tests that 'failed' they're all just failures. (Martin Pool)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

# bzr 2.0.5

Codename:

**2.0.5**: NOT RELEASED YET

#### **Bug Fixes**

- Avoid malloc(0) in patiencediff, which is non-portable. (Martin Pool, #331095)
- Concurrent autopacking is more resilient to already-renamed pack files. If we find that a file we are about to obsolete is already obsoleted, we do not try to rename it, and we leave the file in obsolete\_packs. The code is also fault tolerant if a file goes missing, assuming that another process already removed the file. (John Arbash Meinel, Gareth White, #507557)
- Cope with the lockdir held/info file being empty, which seems to happen fairly often if the process is suddenly interrupted while taking a lock. (Martin Pool, #185103)
- Give the warning about potentially slow cross-format fetches much earlier on in the fetch operation. Don't show this message during upgrades, and show the correct format indication for remote repositories. (Martin Pool, #456077, #515356, #513157)
- Handle renames correctly when there are files or directories that differ only in case. (Chris Jones, Martin Pool, #368931)
- Fixed CHM generation by moving the NEWS section template into a separate file. (lan Clatworthy, #524184)
- If bzr push --create-prefix triggers an unexpected NoSuchFile error, report that error rather than failing with an unhelpful UnboundLocalError. (Andrew Bennetts, #423563)
- Running bzr command without any arguments now shows bzr version number along with rest of the help text. (Parth Malwankar, #369501)
- Use osutils.O\_NOINHERIT for some files on win32 to avoid PermissionDenied errors. (Inada Naoki, #524560)

### Documentation

• Added location-alias help topic. (Andrew Bennetts, #337834)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

### bzr 2.0.4

**Codename:** smooth sailing **2.0.4:** 2010-01-21

The fourth bugfix-only release in the 2.0 series contains more than a dozen bugfixes relative to 2.0.3. The primary focus is on handling interruptions and concurrent operations more cleanly, there is also a fair improvement to bzr export when exporting a remote branch.

#### **Bug Fixes**

- bzr annotate on another branch with -r branch:... no longer fails with an ObjectNotLocked error. (Andrew Bennetts, #496590)
- bzr export dir now requests all file content as a record stream, rather than requsting the file content one file-at-a-time. This can make exporting over the network significantly faster (54min => 9min in one case). (John Arbash Meinel, #343218)
- bzr serve no longer slowly leaks memory. The compiled bzrlib.bencode.Encoder() class was using \_\_del\_\_ to cleanup and free resources, and it should have been using \_\_dealloc\_\_. This will likely have an impact on any other process that is serving for an extended period of time. (John Arbash Meinel, #494406)
- Check for SIGINT (Ctrl-C) and other signals immediately if readdir returns EINTR by calling PyErr\_CheckSignals. This affected the optional \_readdir\_pyx extension. (Andrew Bennetts, #495023)
- Concurrent autopacks will no longer lose a newly created pack file. There was a race condition, where if the reload happened at the right time, the second packer would forget the name of the newly added pack file. (John Arbash Meinel, Gareth White, #507566)
- Give a clearer message if the lockdir disappears after being apparently successfully taken. (Martin Pool, #498378)
- Give a warning when fetching between repositories (local or remote) with sufficiently different formats that the content will need to be serialized (ie InterDifferingSerializer or inventory-deltas), so the user has a clue that upgrading could make it faster. (Martin Pool, #456077)
- If we fail to open ~/.bzr.log write a clear message to stderr

- rather than using warning(). The log file is opened before logging is set up, and it leads to very confusing: 'no handlers for "bzr" messages for users, rather than something nicer. (John Arbash Meinel, Barry Warsaw, #503886)
- Refuse to build with any Pyrex 0.9.4 release, as they have known bugs. (Martin Pool, John Arbash Meinel, #449372)
- setup.py bdist\_rpm now properly finds extra files needed for the build. (there is still the distutils bug http://bugs.python.org/issue644744) (Joe Julian, #175839)
- The 2a format wasn't properly restarting autopacks when something changed underneath it (like another autopack). Now concurrent autopackers will properly succeed. (John Arbash Meinel, #495000)
- TreeTransform can now handle when a delta says that the file id for the tree root changes. Rather than trying to rename your working directory, or failing early saying that you can't have multiple tree roots. This also fixes revert, update, and pull when the root id changes. (John Arbash Meinel, #494269, #504390)
- \_update\_current\_block no longer suppresses exceptions, so ^C at just the right time will get propagated, rather than silently failing to move the block pointer. (John Arbash Meinel, Gareth White, #495023)

### **Testing**

We have a new test\_source that ensures all pyrex cdef functions handle exceptions somehow. (Possibly by setting # cannot\_raise rather than an except ?: clause.) This should help prevent bugs like bug #495023. (John Arbash Meinel)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

### bzr 2.1.0b4

Codename: san francisco airport

**2.1.0b4:** 2009-12-14

The fourth beta release in the 2.1 series brings with it a significant number of bugfixes (~20). The test suite is once again (finally) "green" on Windows, and should remain that way for future releases. There are a few performance related updates (faster upgrade and log), and several UI tweaks. There has also been a significant number of tweaks to the runtime documentation. 2.1.0b4 include everything from the 2.0.3 release.

### **Compatibility Breaks**

 The BZR\_SSH environmental variable may now be set to the path of a secure shell client. If currently set to the value ssh it will now guess the vendor of the program with that name, to restore the old behaviour that indicated the SSH Corporation client use sshcorp instead as the magic string. (Martin <gzlist@googlemail.com>, #176292)

#### **New Features**

- bzr commit now has a --commit-time option. (Alexander Sack, #459276)
- -Dhpss now increases logging done when run on the bzr server, similarly to how it works on the client. (John Arbash Meinel)
- New option bzr unshelve --keep applies the changes and leaves them on the shelf. (Martin Pool, Oscar Fuentes, #492091)
- The BZR\_COLUMNS environment variable can be set to force bzr to respect a given terminal width. This can be useful when output is redirected or in obscure cases where the default value is not appropriate. Pagers can use it to get a better control of the line lengths. (Vincent Ladeuil)
- The new command bzr lp-mirror will request that Launchpad update its mirror of a local branch. This command will only function if launchpadlib is installed. (Jonathan Lange)

#### **Bug Fixes**

- After renaming a file, the dirstate could accidentally reference source\\path rather than source/path on Windows. This might be a source of some dirstate-related failures. (John Arbash Meinel)
- bzr commit now detects commit messages that looks like file names and issues a warning. (Gioele Barabucci, #73073)
- bzr ignore / no longer causes an IndexError. (Gorden Tyler, #456036)
- bzr log -no -rN should not return revisions beyond its merged revisions. (#325618, #484109, Marius Kruger)
- bzr merge --weave and --lca will now create .BASE files for files with conflicts (similar to --merge3). The contents of the file is a synthesis of all bases used for the merge. (John Arbash Meinel, #40412)
- bzr mv --quiet really is quiet now. (Gordon Tyler, #271790)
- bzr serve is more clear about the risk of supplying –allowwrites. (Robert Collins, #84659)
- bzr serve --quiet really is quiet now. (Gordon Tyler, #252834)
- Fix bug with redirected URLs over authenticated HTTP. (Glen Mailer, Neil Martinsen-Burrell, Vincent Ladeuil, #395714)
- Interactive merge doesn't leave branch locks behind. (Aaron Bentley)
- Lots of bugfixes for the test suite on Windows. We should once again have a test suite with no failures on Windows. (John Arbash Meinel)
- osutils.terminal\_width() obeys the BZR\_COLUMNS environment variable but returns None if the terminal is not a tty (when output is redirected for example). Also fixes its usage under OSes that doesn't provide termios.TIOCGWINSZ. Make sure the corresponding tests runs on windows too. (Joke de

- Buhr, Vincent Ladeuil, #353370, #62539) (John Arbash Meinel, Vincent Ladeuil, #492561)
- Terminate ssh subprocesses when no references to them remain, fixing subprocess and file descriptor leaks. (Andrew Bennetts, #426662)
- The --hardlink option of bzr branch and bzr checkout now works for 2a format trees. Only files unaffected by content filters will be hardlinked. (Andrew Bennetts, #408193)
- The new glob expansion on Windows would replace all N characters with 7 even if it there wasn't a glob to expand, the arg was quoted, etc. Now only change slashes if there is something being glob expanded. (John Arbash Meinel, #485771)
- Use our faster KnownGraph.heads() functionality when computing the new rich-root heads. This can cut a conversion time in half (mysql from 13.5h => 6.2h) (John Arbash Meinel, #487632)
- When launching a external diff tool via bzr diff –using, temporary files are no longer created, rather, the path to the file in the working tree is passed to the external diff tool. This allows the file to be edited if the diff tool provides for this. (Gary van der Merwe, #490738)
- The launchpad-open command can now be used from a subdirectory of a branch, not just from the root of the branch. (Neil Martinsen-Burrell, #489102)

#### **Improvements**

- bzr log is now faster. (lan Clatworthy)
- bzr update provides feedback on which branch it is up to date with. (Neil Martinsen-Burrell)
- bzr upgrade from pre-2a to 2a can be significantly faster (4x). For details see the xml8 patch and heads() improvements. (John Arbash Meinel)
- bzrlib.urlutils.local\_path\_from\_url
   'file://localhost/' as well as 'file:///' URLs on POSIX. (Michael Hudson)
- The progress bar now shows only a spinner and per-operation counts, not an overall progress bar. The previous bar was often not correlated with real overall operation progress, either because the operations take nonlinear time, or because at the start of the operation Bazaar couldn't estimate how much work there was to do. (Martin Pool)

### Documentation

• Lots of documentation tweaks for inline help topics and command help information.

# **API Changes**

- bzrlib.textui (vestigial module) removed. (Martin Pool)
- The Launchpad plugin now has a function login which will log in to Launchpad with launchpadlib, and load\_branch which will return the Launchpad Branch object corresponding to a given Bazaar Branch object. (Jonathan Lange)

#### **Internals**

- New test Feature: ModuleAvailableFeature. It is designed to make it easier to handle what tests you want to run based on what modules can be imported. (Rather than lots of customimplemented features that were basically copy-and-pasted.) (John Arbash Meinel)
- osutils.timer\_func() can be used to get either time.time() or time.clock() when you want to do performance timing. time.time() is limited to 15ms resolution on Windows, but time.clock() gives CPU and not wall-clock time on other platforms. (John Arbash Meinel)
- Several code paths that were calling Transport.get().read() have been changed to the equalivent Transport.get\_bytes(). The main difference is that the latter will explicitly call file.close(), rather than expecting the garbage collector to handle it. This helps with some race conditions on Windows during the test suite and sftp tests. (John Arbash Meinel)

# **Testing**

• TestCaseWithMemoryTransport no longer sets \$HOME and \$BZR\_HOME to unicode strings. (Michael Hudson, #464174)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

# bzr 2.0.3

**Codename:** little italy **2.0.3:** 2009-12-14

The third stable release of Bazaar has a small handful of bugfixes. As expected, this has no internal or external compatibility changes versus 2.0.2 (or 2.0.0).

### **Bug Fixes**

- bzr push --use-existing-dir no longer crashes if the directory exists but contains an invalid .bzr directory. (Andrew Bennetts, #423563)
- Content filters are now applied correctly after pull, merge and switch. (lan Clatworthy, #385879)
- Fix a potential segfault in the groupcompress hash map handling code. When inserting new entries, if the final hash bucket was empty, we could end up trying to access if (last\_entry+1)->ptr == NULL. (John Arbash Meinel, #490228)
- Improve "Binary files differ" hunk handling. (Aaron Bentley, #436325)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release

Notes »

previous | next

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

## bzr 2.1.0b3

**Codename:** after sprint recovery

**2.1.0b3**: 2009-11-16

This release was pushed up from its normal release cycle due to a regression in python 2.4 compatibility in 2.1.0b2. Since this regression was caught before 2.1.0b2 was officially announced, the full changelog includes both 2.1.0b3 and 2.1.0b2 changes.

Highlights of 2.1.0b3 are: new globbing code for all commands on Windows, the test suite now conforms to python's trunk enhanced semantics (skip, etc.), and bzr info -v will now report the correct branch and repo formats for Remote objects.

## **New Features**

• Users can define a shelve editor to provide shelf functionality at a granularity finer than per-patch-hunk. (Aaron Bentley)

### **Bug Fixes**

- Fix for shell completion and short options. (Benoît PIERRE)
- Fix bzr --profile-imports with Python 2.6. (Martin Pool)
- Hooks daughter classes should always call the base constructor. (Alexander Belchenko, Vincent Ladeuil, #389648)
- Improve "Binary files differ" hunk handling. (Aaron Bentley, #436325)
- On Windows, do glob expansion at the command-line level (as is usually done in bash, etc.) This means that *all* commands get glob expansion (bzr status, bzr add, bzr mv, etc). It uses a custom command line parser, which allows us to know if a given section was quoted. It means you can now do bzr ignore "\*.py". (John Arbash Meinel, #425510, #426410, #194450)
- Sanitize commit messages that come in from the '-m' flag. We translate 'rn' => 'n' and a plain 'r' => 'n'. The storage layer doesn't allow those because XML store silently translate it anyway. (The parser auto-translates rn => n in ways that are hard for us to catch.)
- Show correct branch and repository format descriptions in bzr info -v on a smart server location. (Andrew Bennetts, #196080)
- The fix for bug #186920 accidentally broke compatibility with python 2.4. (Vincent Ladeuil, #475585)
- Using Repository.get\_commit\_builder().record\_iter\_changes()
  now correctly sets self.inv\_sha1 to a sha1 string and
  self.new\_inventory to an Inventory instance after calling
  self.finish\_inventory(). (Previously it accidently set both
  values as a tuple on self.inv\_sha1. This was missed because
  repo.add\_revision ignores the supplied inventory sha1 and
  recomputes the sha1 from the repo directly. (John Arbash
  Meinel)
- Shelve command refuse to run if there is no real terminal.

(Alexander Belchenko)

• Avoid unnecessarily flushing of trace file; it's now unbuffered at the Python level. (Martin Pool)

### Documentation

- Include Japanese translations for documentation (Inada Naoki)
- New API ui\_factory.make\_output\_stream to be used for sending bulk (rather than user-interaction) data to stdout. This automatically coordinates with progress bars or other terminal activity, and can be overridden by GUIs. (Martin Pool, 493944)

## Internals

• Some of the core groupcompress functionality now releases the GIL before operation. Similar to how zlib and bz2 operate without the GIL in the core compression and decompression routines. (John Arbash Meinel)

## **Testing**

Notes »

- -Dhpssvfs will now trigger on RemoteBzrDir.\_ensure\_real, providing more debugging of VFS access triggers. (Robert Collins)
- KnownFailure is now signalled to ExtendedTestResult using the same method that Python 2.7 uses - addExpectedFailure. (Robert Collins)
- --parallel=fork is now compatible with -subunit. (Robert Collins, Vincent Ladeuil, #419776)
- Reporting of failures shows test ids not descriptions and thus shows parameterised tests correctly. (Robert Collins)
- TestNotApplicable is now handled within the TestCase.run method rather than being looked for within ExtendedTestResult.addError. This provides better handling with other TestResult objects, degrading to sucess rather than error. (Robert Collins)
- The private method \_testConcluded on ExtendedTestResult has been removed it was empty and unused. (Robert Collins)
- UnavailableFeature is now handled within the TestCase.run method rather than being looked for within addError. If the Result object does not have an addNotSupported method, addSkip is attempted instead, and failing that addSuccess. (Robert Collins)
- When a TestResult does not have an addSkip method, skipped tests are now reported as successful tests, rather than as errors.
   This change is to make it possible to get a clean test run with a less capable TestResult. (Robert Collins)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

### bzr 2.1.0b2

Codename: a load off my mind

**2.1.0b2:** 2009-11-02

This is our second feature-filled release since 2.0, pushing us down the path to a 2.1.0. Once again, all bugfixes in 2.0.2 are present in 2.1.0b2.

Key highlights in this release are: improved handling of failures-during-cleanup for commit, fixing a long-standing bug with bzr+http and shared repositories, all 1p: urls to be resolved behind proxies, and a new StaticTuple datatype, allowing us to reduce memory consumption (50%) and garbage collector overhead (40% faster) for many operations.

A new --concurrency option has been added as well as an associated BZR\_CONCURRENCY environment variable to specify the number of processes that can be run concurrently when running bzr selftest. The command-line option overrides the environment variable if both are specified. If none is specified, the number of processes is obtained from the OS as before. (Matt Nordhoff, Vincent Ladeuil)

### **Bug Fixes**

- bzr+http servers no longer give spurious jail break errors when serving branches inside a shared repository. (Andrew Bennetts, #348308)
- Errors during commit are handled more robustly so that knockon errors are less likely to occur, and will not obscure the original error if they do occur. This fixes some causes of TooManyConcurrentRequests and similar errors. (Andrew Bennetts, #429747, #243391)
- Launchpad urls can now be resolved from behind proxies. (Gordon Tyler, Vincent Ladeuil, #186920)
- Reduce the strictness for StaticTuple, instead add a debug flag
   -Dstatic\_tuple which will change apis to be strict and raise errors. This way, most users won't see failures, but developers can improve internals. (John Arbash Meinel, #471193)
- TreeTransform.adjust\_path updates the limbo paths of descendants of adjusted files. (Aaron Bentley)
- Unicode paths are now handled correctly and consistently by the smart server. (Andrew Bennetts, Michael Hudson, #458762)

## **Improvements**

- When reading index files, we now use a staticTuple rather than a plain tuple object. This generally gives a 20% decrease in peak memory, and can give a performance boost up to 40% on large projects. (John Arbash Meinel)
- Peak memory under certain operations has been reduced significantly. (eg, 'bzr branch launchpad standalone' is cut in half) (John Arbash Meinel)

## Documentation

• Filtered views user documentation upgraded to refer to format 2a instead of pre-2.0 formats. (Ian Clatworthy)

# **API Changes**

- Remove deprecated CLIUIFactory. (Martin Pool)
- UIFactory now has new show\_error, show\_message and show\_warning methods, which can be hooked by non-text UIs. (Martin Pool)

#### **Internals**

- Added bzrlib.\_simple\_set\_pyx. This is a hybrid between a Set and a Dict (it only holds keys, but you can lookup the object located at a given key). It has significantly reduced memory consumption versus the builtin objects (1/2 the size of Set, 1/3rd the size of Dict). This is used as the interning structure for StaticTuple objects. (John Arbash Meinel)
- bzrlib.\_static\_tuple\_c.staticTuple is now available and used by the btree index parser and the chk map parser. This class functions similarly to tuple objects. However, it can only point to a limited collection of types. (Currently StaticTuple, str, unicode, None, bool, int, long, float, but not subclasses). This allows us to remove it from the garbage collector (it cannot be in a cycle), it also allows us to intern the objects. In testing, this can reduce peak memory by 20-40%, and significantly improve performance by removing objects from being inspected by the garbage collector. (John Arbash Meinel)
- GroupCompressBlock.\_ensure\_content() will now release the zlib.decompressobj() when the first request is for all of the content. (Previously it would only be released if you made a request for part of the content, and then all of it later.) This turns out to be a significant memory savings, as a zstream carries around approx 260kB of internal state and buffers. (For branching bzr.dev this drops peak memory from 382MB => 345MB.) (John Arbash Meinel)
- When streaming content between 2a format repositories, we now clear caches from earlier versioned files. (So 'revisions' is cleared when we start reading 'inventories', etc.) This can have a significant impact on peak memory for initial copies (~200MB). (John Arbash Meinel)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

# bzr 2.0.2

**Codename:** after the scare **2.0.2:** 2009-11-02

The second in our "let's keep the stable bugfixes flowing" series. As expected this has a few (~9) bugfixes relative to 2.0.1, and no major api changes or features.

### **Bug Fixes**

- Avoid "NoneType has no attribute st\_mode" error when files disappear from a directory while it's being read. (Martin Pool, #446033)
- Content filters are now applied correctly after revert. (lan Clatworthy)
- Diff parsing handles "Binary files differ" hunks. (Aaron Bentley, #436325)
- Fetching from stacked pre-2a repository via a smart server no longer fails intermittently with "second push failed to complete". (Andrew Bennetts, #437626)
- Fix typos left after test\_selftest refactoring. (Vincent Ladeuil, Matt Nordhoff, #461149)
- Fixed ObjectNotLocked errors during bzr log -r NNN somefile. (Andrew Bennetts, #445171)
- PreviewTree file names are not limited by the encoding of the temp directory's filesystem. (Aaron Bentley, #436794)

# **Improvements**

• bzr log now read-locks branches exactly once, so makes better use of data caches. (Andrew Bennetts)

## Documentation

• Filtered views user documentation upgraded to refer to format 2a instead of pre-2.0 formats. (lan Clatworthy)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

### bzr 2.1.0b1

Codename: While the cat is away

**2.1.0b1:** 2009-10-14

This is the first development release in the new split "stable" and "development" series. As such, the release is a snapshot of bzr.dev without creating a release candidate first. This release includes a fair amount of internal changes, with deprecated code being removed, and several new feature developments. People looking for a stable code base with only bugfixes should focus on the 2.0.1 release. All bugfixes present in 2.0.1 are present in 2.1.0b1.

Highlights include support for bzr+ssh://host/~/homedir style urls, finer control over the plugin search path via extended BZR\_PLUGIN\_PATH syntax, visible warnings when extension modules fail to load, and improved error handling during unlocking.

#### **New Features**

- Bazaar can now send mail through Apple OS X Mail.app. (Brian de Alwis)
- bzr+ssh and bzr paths can now be relative to home directories specified in the URL. Paths starting with a path segment of are relative to the home directory of the user running the server, and paths starting with ~user are relative to the home directory of the named user. For example, for a user "bob" with a home directory of /home/bob, these URLs are all equivalent:
  - o bzr+ssh://bob@host/~/repo
  - o bzr+ssh://bob@host/~bob/repo
  - o bzr+ssh://bob@host/home/bob/repo

If bzr serve was invoked with a --directory argument, then no home directories outside that directory will be accessible via this method.

This is a feature of bzr serve, so pre-2.1 clients will automatically benefit from this feature when bzr on the server is upgraded. (Andrew Bennetts, #109143)

- Extensions can now be compiled if either Cython or Pyrex is available. Currently Pyrex is preferred, but that may change in the future. (Arkanes)
- Give more control on BZR\_PLUGIN\_PATH by providing a way to refer to or disable the user, site and core plugin directories. (Vincent Ladeuil, #412930, #316192, #145612)

### **Bug Fixes**

- Bazaar's native protocol code now correctly handles EINTR, which most noticeably occurs if you break in to the debugger while connected to a bzr+ssh server. You can now can continue from the debugger (by typing 'c') and the process continues. However, note that pressing C-in the shell may still kill the SSH process, which is bug 162509, so you must sent a signal to the bzr process specifically, for example by typing kill -QUIT PID in another shell. (Martin Pool, #341535)
- bzr add in a tree that has files with \r or \n in the filename will issue a warning and skip over those files. (Robert Collins, #3918)
- bzr dpush now aborts if uncommitted changes (including pending merges) are present in the working tree. The configuration option dpush\_strict can be used to set the default for this behavior. (Vincent Ladeuil, #438158)
- bzr merge and bzr remove-tree now requires —force if pending merges are present in the working tree. (Vincent Ladeuil, #426344)
- Clearer message when Bazaar runs out of memory, instead of a MemoryError traceback. (Martin Pool, #109115)
- Don't give a warning on Windows when failing to import \_readdir\_pyx as it is never built. (John Arbash Meinel, #430645)
- Don't restrict the command name used to run the test suite.
   (Vincent Ladeuil, #419950)
- ftp transports were built differently when the kerberos python module was present leading to obscure failures related to ASCII/BINARY modes. (Vincent Ladeuil, #443041)
- Network streams now decode adjacent records of the same type into a single stream, reducing layering churn. (Robert Collins)
- PreviewTree behaves correctly when get\_file\_mtime is invoked

- on an unmodified file. (Aaron Bentley, #251532)
- Registry objects should not use iteritems() when asked to use items(). (Vincent Ladeuil, #430510)
- Weave based repositories couldn't be cloned when committers were using domains or user ids embedding '.sig'. Now they can. (Matthew Fuller, Vincent Ladeuil, #430868)

### **Improvements**

- Revision specifiers can now be given in a more DWIM form, without needing explicit prefixes for specifiers like tags or revision id's. See bzr help revisionspec for full details. (Matthew Fuller)
- Bazaar gives a warning before exiting, and writes into .bzr.log, if compiled extensions can't be loaded. This typically indicates a packaging or installation problem. In this case Bazaar will keep running using pure-Python versions, but this may be substantially slower. The warning can be disabled by setting ignore\_missing\_extensions = True in bazaar.conf. See also <a href="https://answers.launchpad.net/bzr/+faq/703">https://answers.launchpad.net/bzr/+faq/703</a>. (Martin Pool, #406113, #430529)
- Secondary errors that occur during Branch.unlock and Repository.unlock no longer obscure the original error. These methods now use a new decorator, only\_raises. This fixes many causes of TooManyConcurrentRequests and similar errors. (Andrew Bennetts, #429747)

## Documentation

- Describe the new shell-like test feature. (Vincent Ladeuil)
- Help on hooks no longer says 'Not deprecated' for hooks that are currently supported. (lan Clatworthy, #422415)

### **API Changes**

- bzrlib.user\_encoding has been removed; use
   bzrlib.osutils.get\_user\_encoding instead. (Martin Pool)
- bzrlib.tests now uses stopTestRun for its TestResult subclasses the same as python's unittest module. (Robert Collins)
- diff.\_get\_trees\_to\_diff
   has been renamed to diff.get\_trees\_and\_branches\_to\_diff. It is now a public API, and it returns the old and new branches. (Gary van der Merwe)
- bzrlib.trace.log\_error, error and info have been deprecated.
   (Martin Pool)
- MutableTree.has\_changes() does not require a tree parameter anymore. It now defaults to comparing to the basis tree. It now checks for pending merges too. Merger.check\_basis has been deprecated and replaced by the corresponding has\_changes() calls. Merge.compare\_basis, Merger.file\_revisions and Merger.ensure\_revision\_trees have also been deprecated. (Vincent Ladeuil, #440631)
- ProgressTask.note is deprecated. (Martin Pool)

#### **Internals**

- Added -Drelock debug flag. It will note a message every time a repository or branch object is unlocked then relocked the same way. (Andrew Bennetts)
- BTreeLeafParser.extract\_key has been tweaked slightly to reduce mallocs while parsing the index (approx 3=>1 mallocs per key read). This results in a 10% speedup while reading an index. (John Arbash Meinel)
- The bzrlib.lsprof module has a new class BzrProfiler which makes profiling in some situations like callbacks and generators easier. (Robert Collins)

## **Testing**

- Passing --1sprof-tests -v to bzr selftest will cause Isprof output to be output for every test. Note that this is very verbose! (Robert Collins)
- Setting BZR\_TEST\_PDB=1 when running selftest will cause a pdb post\_mortem to be triggered when a test failure occurs. (Robert Collins)
- Shell-like tests can now be written. Code in bzrlib/tests/script.py , documentation in developers/testing.txt for details. (Vincent Ladeuil)
- Some tests could end up with the same id, that was dormant for a long time. (Vincent Ladeuil, #442980)
- Stop showing the number of tests due to missing features in the test progress bar. (Martin Pool)
- Test parameterisation now does a shallow copy, not a deep copy of the test to be parameterised. This is not expected to break external use of test parameterisation, and is substantially faster. (Robert Collins)
- Tests that try to open a bzr dir on an arbitrary transport will now
  fail unless they have explicitly permitted the transport via
  self.permit\_url. The standard test factories such as
  self.get\_url will permit the urls they provide automatically, so
  only exceptional tests should need to do this. (Robert Collins)
- The break-in test no longer cares about clean shutdown of the child, instead it is happy if the debugger starts up. (Robert Collins)
- The full test suite is expected to pass when the C extensions are not present. (Vincent Ladeuil, #430749)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

# bzr 2.0.1

**Codename:** Stability First 2.0.1: 2009-10-14

The first of our new ongoing bugfix-only stable releases has arrived. It includes a collection of 12 bugfixes applied to bzr 2.0.0, but does not include any of the feature development in the 2.1.0 series.

### **Bug Fixes**

- bzr add in a tree that has files with \r or \n in the filename will issue a warning and skip over those files. (Robert Collins, #3918)
- bzr will attempt to authenticate with SSH servers that support keyboard-interactive auth but not password auth when using Paramiko. (Andrew Bennetts, #433846)
- Fixed fetches from a stacked branch on a smart server that were failing with some combinations of remote and local formats. This was causing "unknown object type identifier 60" errors. (Andrew Bennetts, #427736)
- Fixed ObjectNotLocked errors when doing some log and diff operations on branches via a smart server. (Andrew Bennetts, #389413)
- Handle things like bzr add foo and bzr rm foo when the tree is at the root of a drive. osutils.\_cicp\_canonical\_relpath always assumed that abspath() returned a path that did not have a trailing /, but that is not true when working at the root of the filesystem. (John Arbash Meinel, Jason Spashett, #322807)
- Hide deprecation warnings for 'final' releases for python2.6. (John Arbash Meinel, #440062)
- Improve the time for bzr log DIR for 2a format repositories. We had been using the same code path as for <2a formats, which required iterating over all objects in all revisions. (John Arbash Meinel, #374730)</li>
- Make sure that we unlock the tree if we fail to create a TreeTransform object when doing a merge, and there is limbo, or pending-deletions directory. (Gary van der Merwe, #427773)
- Occasional IndexError on renamed files have been fixed.
   Operations that set a full inventory in the working tree will now go via the apply inventory delta code path which is simpler and

easier to understand than dirstates set\_state\_from\_inventory method. This may have a small performance impact on operations built on \_write\_inventory, but such operations are already doing full tree scans, so no radical performance change should be observed. (Robert Collins, #403322)

- Retrieving file text or mtime from a \_PreviewTree has good performance when there are many changes. (Aaron Bentley)
- The CHK index pages now use an unlimited cache size. With a limited cache and a large project, the random access of chk pages could cause us to download the entire cix file many times. (John Arbash Meinel, #402623)
- When a file kind becomes unversionable after being added, a sensible error will be shown instead of a traceback. (Robert Collins, #438569)

### Documentation

- Improved README. (Ian Clatworthy)
- Improved upgrade documentation for Launchpad branches. (Barry Warsaw)

#### bzr 2.0.0

**2.0.0:** 2009-09-22 **Codename:** Instant Karma

This release of Bazaar makes the 2a (previously 'brisbane-core') format the default when new branches or repositories are created. This format is substantially smaller and faster for many operations. Most of the work in this release focuses on bug fixes and stabilization, covering both 2a and previous formats. (See the Upgrade Guide for information on migrating existing projects.)

This release also improves the documentation content and presentation, including adding Windows HtmlHelp manuals.

The Bazaar team decided that 2.0 will be a long-term supported release, with bugfix-only 2.0.x releases based on it, continuing for at least six months or until the following stable release.

## Changes from 2.0.0rc2 to final

 Officially branded as 2.0.0 rather than 2.0 to clarify between things that "want to happen on the 2.0.x stable series" versus things that want to "land in 2.0.0". (Changes how bzrlib.\_format\_version\_tuple() handles micro = 0.) (John Arbash Meinel)

previous | next

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release

Notes »

# bzr 2.0.0rc2

**2.0.0rc2:** 2009-09-10

### **New Features**

 Added post\_commit hook for mutable trees. This allows the keywords plugin to expand keywords on files changed by the commit. (lan Clatworthy, #408841)

#### **Bug Fixes**

- Bazaar's native protocol code now correctly handles EINTR, which most noticeably occurs if you break in to the debugger while connected to a bzr+ssh server. You can now can continue from the debugger (by typing 'c') and the process continues. However, note that pressing C-in the shell may still kill the SSH process, which is bug 162509, so you must sent a signal to the bzr process specifically, for example by typing kill -QUIT PID in another shell. (Martin Pool, #341535)
- bzr check in pack-0.92, 1.6 and 1.9 format repositories will no longer report incorrect errors about Missing inventory ('TREE\_ROOT', ...) (Robert Collins, #416732)
- bzr info -v on a 2a format still claimed that it was a "Development format" (John Arbash Meinel, #424392)
- bzr log stacked-branch shows the full log including revisions that are in the fallback repository. (Regressed in 2.0rc1). (John Arbash Meinel, #419241)
- Clearer message when Bazaar runs out of memory, instead of a MemoryError traceback. (Martin Pool, #109115)
- Conversion to 2a will create a single pack for all the new revisions (as long as it ran without interruption). This improves both bzr upgrade and bzr pull or bzr merge from local branches in older formats. The autopack logic that occurs every 100 revisions during local conversions was not returning that pack's identifier, which resulted in the partial packs created during the conversion not being consolidated at the end of the conversion process. (Robert Collins, #423818)
- Fetches from 2a to 2a are now again requested in 'groupcompress' order. Groups that are seen as 'underutilized' will be repacked on-the-fly. This means that when the source is fully packed, there is minimal overhead during the fetch, but if

- the source is poorly packed the result is a fairly well packed repository (not as good as 'bzr pack' but good-enough.) (Robert Collins, John Arbash Meinel, #402652)
- Fix a potential segmentation fault when doing 'log' of a branch that had ghosts in its mainline. (Evaluating None as a tuple is bad.) (John Arbash Meinel, #419241)
- groupcompress sort order is now more stable, rather than relying on topo\_sort ordering. The implementation is now KnownGraph.gc\_sort. (John Arbash Meinel)
- Local data conversion will generate correct deltas. This is a critical bugfix vs 2.0rc1, and all 2.0rc1 users should upgrade to 2.0rc2 before converting repositories. (Robert Collins, #422849)
- Network streams now decode adjacent records of the same type into a single stream, reducing layering churn. (Robert Collins)
- Prevent some kinds of incomplete data from being committed to a 2a repository, such as revisions without inventories, a missing chk\_bytes record for an inventory, or a missing text referenced by an inventory. (Andrew Bennetts, #423506, #406687)

#### **Documentation**

- Fix assertion error about "\_remember\_remote\_is\_before" when pushing to older smart servers. (Andrew Bennetts, #418931)
- Help on hooks no longer says 'Not deprecated' for hooks that are currently supported. (lan Clatworthy, #422415)
- PDF and CHM (Windows HtmlHelp) formats are now supported for the user documentation. The HTML documentation is better broken up into topics. (Ian Clatworthy)
- The developer and foreign language documents are now separated out so that searching in the HTML and CHM files produces more useful results. (Ian Clatworthy)
- The main table of contents now provides links to the new Migration Docs and Plugins Guide. (Ian Clatworthy)

# bzr 2.0.0rc1

**Codename:** no worries **2.0.0rc1:** 2009-08-26

#### **Compatibility Breaks**

- The default format for bzr is now 2a. This format brings many significant performance and size improvements. bzr can pull from any existing repository into a 2a one, but can only transfer from 2a into rich-root repositories. The Upgrade guide has more information about this change. (Robert Collins)
- On Windows auto-detection of Putty's plink.exe is disabled.
   Default SSH client for Windows is paramiko. User still can force usage of plink if explicitly set environment variable BZR\_SSH=plink. (#414743, Alexander Belchenko)

#### **New Features**

• bzr branch --switch can now switch the checkout in the current directory to the newly created branch. (Lukáš Lalinský)

#### **Bug Fixes**

- Further tweaks to handling of bzr add messages about ignored files. (Jason Spashett, #76616)
- Fetches were being requested in 'groupcompress' order, but weren't recombining the groups. Thus they would 'fragment' to get the correct order, but not 'recombine' to actually benefit from it. Until we get recombining to work, switching to 'unordered' fetches avoids the fragmentation. (John Arbash Meinel, #402645)
- Fix a pycurl related test failure on karmic by recognizing an error raised by newer versions of pycurl. (Vincent Ladeuil, #306264)
- Fix a test failure on karmic by making a locale test more robust. (Vincent Ladeuil, #413514)
- Fix IndexError printing CannotBindAddress errors. (Martin Pool, #286871)
- Fix "Revision ... not present" errors when upgrading stacked branches, or when doing fetches from a stacked source to a stacked target. (Andrew Bennetts, #399140)
- bzr branch of 2a repositories over HTTP is much faster. bzr now batches together small fetches from 2a repositories, rather than fetching only a few hundred bytes at a time. (Andrew Bennetts, #402657)

#### **Improvements**

- A better description of the platform is shown in crash tracebacks, bzr --version and bzr selftest. (Martin Pool, #409137)
- bzr can now (again) capture crash data through the apport library, so that a single human-readable file can be attached to bug reports. This can be disabled by using -Dno\_apport on the command line, or by putting no\_apport into the debug\_flags section of bazaar.conf. (Martin Pool, Robert Collins, #389328)
- bzr push locally on windows will no longer give a locking error with dirstate based formats. (Robert Collins)
- bzr shelve and bzr unshelve now work on windows. (Robert Collins, #305006)
- Commit of specific files no longer prevents using the iter\_changes codepath. On 2a repositories, commit of specific files should now be as fast, or slightly faster, than a full commit. (Robert Collins)
- The internal core code that handles specific file operations like bzr st filename or bzr commit filename has been changed to include the parent directories if they have altered, and when a directory stops being a directory its children are always included. This fixes a number of causes for InconsistentDelta errors, and permits faster commit of specific paths. (Robert Collins, #347649)

## Documentation

• New developer documentation for content filtering. (Martin Pool)

## **API Changes**

 bzrlib.shelf\_ui has had the from\_args convenience methods of its classes changed to manage lock lifetime of the trees they open in a way consistent with reader-exclusive locks. (Robert Collins, #305006)

# **Testing**

### bzr 1.18.1

**Codename:** nein nein nein! **1.18.1:** 2009-09-09

This release fixes two small but worthwhile bugs relevant to users on Microsoft Windows: some commands that failed on with locking errors will now work, and a bug that caused poor performance after committing a file with line-ending conversion has now been fixed. It also fixes a bug in pushing to older servers.

## **Bug Fixes**

- Fixed a problem where using content filtering and especially end-of-line conversion will commit too many copies a file. (Martin Pool, #415508)
- Fix assertion error about \_remember\_remote\_is\_before in set\_tags\_bytes when pushing to older smart servers. (Andrew Bennetts, Alexander Belchenko, #418931)

## **Improvements**

- bzr push locally on Windows will no longer give a locking error with dirstate based formats. (Robert Collins)
- bzr shelve and bzr unshelve now work on Windows. (Robert Collins, #305006)

#### **API Changes**

- bzrlib.shelf\_ui has had the from\_args convenience methods of its classes changed to manage lock lifetime of the trees they open in a way consistent with reader-exclusive locks. (Robert Collins, #305006)
- Tree.path\_content\_summary may return a size of None, when called on a tree with content filtering where the size of the canonical form cannot be cheaply determined. (Martin Pool)
- When manually creating transport servers in test cases, a new helper TestCase.start\_server that registers a cleanup and starts the server should be used. (Robert Collins)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release Notes »

previous | next

# bzr 1.18

### **Compatibility Breaks**

 Committing directly to a stacked branch from a lightweight checkout will no longer work. In previous versions this would appear to work but would generate repositories with insufficient data to create deltas, leading to later errors when branching or reading from the repository. (Robert Collins, bug #375013)

## New Features

#### **Bug Fixes**

- Fetching from 2a branches from a version-2 bzr protocol would fail to copy the internal inventory pages from the CHK store. This cannot happen in normal use as all 2a compatible clients and servers support the version-3 protocol, but it does cause test suite failures when testing downlevel protocol behaviour. (Robert Collins)
- Fix a test failure on karmic by making a locale test more robust. (Vincent Ladeuil, #413514)
- Fixed "Pack ... already exists" error when running bzr pack on a fully packed 2a repository. (Andrew Bennetts, #382463)
- Further tweaks to handling of bzr add messages about ignored files. (Jason Spashett, #76616)
- Properly handle fetching into a stacked branch while converting the data, especially when there are also ghosts. The code was filling in parent inventories incorrectly, and also not handling when one of the parents was a ghost. (John Arbash Meinel, #402778, #412198)
- RemoteStreamSource.get\_stream\_for\_missing\_keys
   CHK inventory pages when appropriate (by falling back to the vfs stream source). (Andrew Bennetts, #406686)
- StreamSource generates rich roots from non-rich root sources correctly now. (Andrew Bennetts, #368921)
- When deciding whether a repository was compatible for upgrading or fetching, we previously incorrectly checked the default repository format for the bzrdir format, rather than the format that was actually present on disk. (Martin Pool, #408824)

#### **Improvements**

- A better description of the platform is shown in crash tracebacks, bzr --version and bzr selftest. (Martin Pool, #409137)
- Cross-format fetches (such as between 1.9-rich-root and 2a) via
  the smart server are more efficient now. They send inventory
  deltas rather than full inventories. The smart server has two new
  requests,
  Repository.get\_stream\_1.19
  and
  Repository.insert\_stream\_1.19
  to support this. (Andrew
  Bennetts, #374738, #385826)
- Extracting the full ancestry and computing the merge\_sort is now significantly faster. This effects things like bzr log -n0. (For example, bzr log -r -10..-1 -n0 bzr.dev is 2.5s down to 1.0s. (John Arbash Meinel)

## Documentation

# **API Changes**

#### **Internals**

- -Dstrict\_locks can now be used to check that read and write locks are treated properly w.r.t. exclusivity. (We don't try to take an OS read lock on a file that we already have an OS write lock on.) This is now set by default for all tests, if you have a test which cannot be fixed, you can use self.thisFailsStrictLockCheck() as a compatibility knob. (John Arbash Meinel)
- InterDifferingSerializer is now only used locally. Other fetches that would have used InterDifferingSerializer now use the more network friendly StreamSource, which now automatically does the same transformations as InterDifferingSerializer. (Andrew Bennetts)
- KnownGraph now has a .topo\_sort and .merge\_sort member which are implemented in pyrex and significantly faster. This is exposed along with CombinedGraphIndex.find\_ancestry() as VersionedFiles.get\_known\_graph\_ancestry(keys). (John Arbash Meinel)
- RemoteBranch.open now honours ignore\_fallbacks correctly on bzr-v2 protocols. (Robert Collins)
- The index code now has some specialized routines to extract the full ancestry of a key in a more efficient manner. CombinedGraphIndex.find\_ancestry(). (Time to get ancestry for bzr.dev drops from 1.5s down to 300ms. For OOo from 33s => 10.5s) (John Arbash Meinel)

## **Testing**

• Install the test ssl certificate and key so that installed bzr can run the https tests. (Denys Duchier, #392401)

# bzr 1.18rc1

**Codename:** little traveller **1.18:** 2009-08-20 **1.18rc1:** 2009-08-10

This release of Bazaar marches on towards the 2.0 release in which the 2a 'brisbane-core' format becomes generally recommended. Most of the work in this release now focusses on bug fixes and stabilization, covering both 2a and previous formats. There is a new text-mode interactive merge feature, a new guide to migration to 2a format in the user documentation, and pushing branches to a smart server is now much faster.

The Bazaar team decided that 2.0 will be a long-term supported release, with bugfix-only releases based on it continuing for at least six months or until the following stable release.

There are no changes from 1.18rc1 to 1.18.

#### **New Features**

- bzr merge --interactive applies a user-selected portion of the merge. The UI is similar to shelve. (Aaron Bentley)
- bzr reconfigure now takes options --stacked-on URL and -unstacked to change stacking of a branch. (Martin Pool, #391411)

### **Bug Fixes**

- Annotating on a stacked branch will now succeed in simple scenarios. There are still some complex scenarios where it will fail (bug #399884) (John Arbash Meinel, #393366)
- A progress bar is no longer left dangling when bzr selftest completes, and the progress bar updates with zero latency so the displayed test name is always the one that's actually running. (Martin Pool, #123688)
- Authenticating against an ssh server now uses auth\_none to determine if password authentication is even supported. This fixes a bug where users would be prompted for a launchpad password, even though launchpad only supports publickey authentication. (John Arbash Meinel, #375867)
- BranchBuilder now accepts timezone to avoid test failures in countries far from GMT. (Vincent Ladeuil, #397716)
- bzr commit no longer saves the unversioning of missing files until the commit has completed on the branch. This means that aborting a commit that found a missing file will leave the tree unedited. (Robert Collins, #282402)
- bzr mv no longer takes out branch locks, which allows it to work when the branch is readonly. (Robert Collins, #216541)
- bzr revert . no longer generates an InconsistentDelta error when there are missing subtrees. (Robert Collins, #367632)
- bzr send now generates valid bundles with --2a formats. However, do to internal changes necessary to support this, older clients will fail when trying to insert them. For newer clients, the bundle can be used to apply the changes to any rich-root compatible format. (John Arbash Meinel, #393349)
- Cope with FTP servers that don't support restart/append by falling back to reading and then rewriting the whole file, such as TahoeLAFS. (This fallback may be slow for some access

- patterns.) (Nils Durner, #294709)
- Encode the paths in mbcs encoding on Windows when spawning an external diff client. This at least allows supporting filenames that are not ascii, but are present in the current locale. Ideally we would be able to pass the Unicode path, but that would be client dependent. (John Arbash Meinel, #382709)
- Fix a compile bug on Solaris having to do with const and pointer-to-pointers. (John Arbash Meinel, #408441)
- Fixed a NameError that occurs when merging or pulling from a URL that causes a redirection loop when bzr tries to read a URL as a bundle. (Andrew Bennetts, #400847)
- Fix AttributeError: 'TestUIFactory' object has no attribute 'tick' running send and similar commands on 2a formats. (Martin Pool, #408201)
- Fix crash in some invocations of bzr status in format 2a. (Martin Pool, #403523)
- Fixed export to existing directory: if directory is empty then export will succeed, otherwise it fails with error. (Alexander Belchenko, #406174)
- Fixed spurious "Source branch does not support stacking" warning when pushing. (Andrew Bennetts, #388908)
- Fixed spurious transport activity indicator appearing while tests are running. (Martin Pool, #343532)
- Merge now correctly handles empty right-hand revision specs. (Aaron Bentley, #333961)
- Renames to lexographically lower basenames in trees that have never been committed to will no longer corrupt the dirstate. This was caused by an bug in the dirstate update\_minimal method. (Robert Collins, #395556)
- Requests for unknown methods no longer cause the smart server to log lots of backtraces about UnknownSmartMethod, do\_chunk or do\_end. (Andrew Bennetts, #338561)
- Shelve will not shelve the initial add of the tree root. (Aaron

Bentley)

- Streaming from bzr servers where there is a chain of stacked branches (A stacked on B stacked on C) will now work. (Robert Collins, #406597)
- The environment variable BZR\_PROGRESS\_BAR set to either text or none always forces progress bars either on or off respectively. Otherwise, they're turned on if TERM is not dumb and stderr is a terminal. bzr always uses the 'text' user interface when run as a command, so BZR\_USE\_TEXT\_UI is no longer needed. (Martin Pool, #339385, #387717)
- The optional \_knit\_load\_data\_pyx C extension was never being imported. This caused significant slowdowns when reading data from repositories. (Andrew Bennetts, #405653)
- There was a bug in osutils.relpath that was only triggered on Windows. Essentially if you were at the root of a drive, and did something to a branch/repo on another drive, we would go into an infinite loop while trying to find a 'relative path'. (John Arbash Meinel, #394227)
- workingTree4.unversion will no longer fail to unversion ids which were present in a parent tree but renamed in the working tree. (Robert Collins, #187207)

# **Improvements**

- Can now rename/move files even if they have been removed from the inventory. (Marius Kruger)
- Pushing branches with tags via bzr:// and bzr+ssh:// is much faster, using a new Branch.set\_tags\_bytes smart server verb rather than VFS methods. For example, pushes of small branches with tags take 11 rather than 18 smart server requests. (Andrew Bennetts, #398608)
- Sending Ctrl-Break on Windows will now drop you into the debugger, in the same way that sending Ctrl-\ does on other platforms. (John Arbash Meinel)

# Documentation

• Added Bazaar 2.0 Upgrade Guide. (Ian Clatworthy)

# **API Changes**

- CLIUIFactory is deprecated; use TextulFactory instead if you need to subclass or create a specific class, or better yet the existing make\_ui\_for\_terminal. SilentulFactory is clarified to do no user interaction at all, rather than trying to read from stdin but not writing any output, which would be strange if reading prompts or passwords. (Martin Pool)
- New TransformPreview.commit() allows committing without a working tree. (Aaron Bentley)
- pb parameter to TextTestResult is deprecated and ignored.
   (Martin Pool)
- ProgressTasks now prefer to talk direct to their ProgressView not to the UIFactory. (Martin Pool)
- WorkingTree.\_check now requires a references dict with keys matching those returned by WorkingTree.\_get\_check\_refs. (Robert Collins)

#### **Internals**

- CHKInventory.path2id uses the parent\_id to basename hash to avoid reading the entries along the path, reducing work to lookup ids from paths. (Robert Collins)
- CHKMap.apply\_delta now raises InconsistentDelta if a delta adds as new a key which was already mapped. (Robert Collins)
- Inventory delta application catches more cases of corruption and can prevent corrupt deltas from affecting consistency of data structures on disk. (Robert Collins)
- –subunit support now adds timestamps if the subunit version supports it. (Robert Collins)
- The Windows all-in-one installer now bundles the PyQt image format plugins, which allows previewing more images as part of 'qdiff'. (Alexander Belchenko)

# **Testing**

- Merge directive cherrypick tests must use the same root id. (Martin Pool, #409684)
- Spurious failure in <a href="https://check.com/check.c
- The bzrlib.tests.TextTestRunner will longer no call countTestsCases on the test being run. Progress information is handled by having the instead test passed in call result.progress before running its contents. This improves the behaviour when using TextTestRunner with test suites that don't support countTestsCases. (Robert Collins)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

# bzr 1.17.1 (unreleased)

# **Bug Fixes**

 The optional \_knit\_load\_data\_pyx C extension was never being imported. This caused significant slowdowns when reading data from knit format repositories. (Andrew Bennetts, #405653)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

# bzr 1.17

Codename: so-late-its-brunch

**1.17rc1**: 2009-07-13 **1.17**: 2009-07-20

Bazaar continues to blaze a straight and shining path to the 2.0 release and the elevation of the 2a beta format to the full glory of "supported and stable".

Highlights in this release include greatly reduced memory consumption during commits, faster ls, faster annotate, faster network operations if you're specifying a revision number and the final destruction of those annoying progress bar artifacts.

# Changes from 1.17rc1 to 1.17final

- Change an extension to call the python frozenset() rather than the C api PyFrozenset\_New. It turns out that python2.4 did not expose the C api. (John Arbash Meinel, #399366)
- Fixes for the Makefile and the rename of <a href="generate\_docs.py">generate\_docs.py</a> to tools/generate\_docs.py to allow everything to be built on Windows. (John Arbash Meinel, #399356)
- bzr serve once again applies a chrootserver to the given directory before serving it. (Andrew Bennetts, #400535)

# **Compatibility Breaks**

• bzr register-branch from the Launchpad plugin now refers to "project" instead of "product" which is the correct Launchpad terminology. The —product option is deprecated and users should switch to using —project. (Neil Martinsen-Burrell, #238764)

#### **New Features**

- bzr push now aborts if uncommitted changes (including pending merges) are present in the working tree (if one is present) and no revision is specified. The configuration option push\_strict can be used to set the default for this behavior. (Vincent Ladeuil, #284038, #322808, #65286)
- bzr revno and bzr revision-info now have a --tree option to show revision info for the working tree instead of the branch. (Matthew Fuller, John Arbash Meinel)
- bzr send now aborts if uncommitted changes (including pending merges) are present in the working tree and no revision is specified. The configuration option send\_strict can be used to set the default for this behavior. (Vincent Ladeuil, #206577)
- bzr switch --create-branch/-b can now be used to create and switch to a new branch. Supplying a name without a / will create the branch relative to the existing branch. (similar to how bzr switch name works when the branch already exists.) (John Arbash Meinel)

#### **Bug Fixes**

- Accept uppercase "Y/N" to prompts such as from break lock. (#335182, Tim Powell, Martin Pool)
- Add documentation about diverged branches and how to fix them in the centralized workflow with local commits. Mention bzr help diverged-branches when a push fails because the branches have diverged. (Neil Martinsen-Burrell, #269477)
- Annotate would sometimes 'latch on' to trivial lines, causing important lines to be incorrectly annotated. (John Arbash Meinel, #387952)
- Automatic format upgrades triggered by default stacking policies on a 1.16rc1 (or later) smart server work again. (Andrew Bennetts, #388675)
- Avoid progress bar artifacts being left behind on the screen.
   (Martin Pool, #321935)
- Better message in bzr split error suggesting a rich root format.
   (Neil Martinsen-Burrell, #220067)
- Branch.set\_append\_revisions\_only now works with branches on a smart server. (Andrew Bennetts, #365865)
- By default, bzr branch will fail if the target directory exists, but does not already have a control directory. The flag --useexisting-dir will allow operation to proceed. (Alexander Belchenko, #307554)
- bzr 1s DIR --from-root now shows only things in DIR, not everything. (lan Clatworthy)
- Fetch between repositories does not error if they have inconsistent data that should be irrelevant to the fetch operation. (Aaron Bentley)
- Fix AttributeError exception when reconfiguring lightweight checkout of a remote repository. (Jelmer Vernooij, #332194)
- Fix bug in decoding v3 smart server messages when receiving

- multiple lots of excess bytes after an end-of-message. (Andrew Bennetts)
- Force deletion of readonly files during merge, update and other tree transforms. (Craig Hewetson, Martin Pool, #218206)
- Force socket shutdown in threaded http test servers to avoid client hangs (pycurl). (Vincent Ladeuil, #383920).
- LRUCache will maintain the linked list pointers even if a nodes cleanup function raises an exception. (John Arbash Meinel, #396838)
- Progress bars are now suppressed again when the environment variable BZR\_PROGRESS\_BAR is set to none. (Martin Pool, #339385)
- Reduced memory consumption during bzr commit of large files.
   For pre 2a formats, should be down to ~3x the size of a file. For --2a format repositories, it is down to the size of the file content plus the size of the compressed text. Related to bug #109114. (John Arbash Meinel)
- Set hidden attribute on .bzr directory below unicode path should never fail with error. The operation should succeed even if bzr unable to set the attribute. (Alexander Belchenko, related to bug #335362).
- Stacking will no longer accept requests to stack on the same branch/repository. Existing branches that incorrectly reference the same repository in a stacking configuration will now raise UnstackableLocationError when the branch is opened. This can be fixed by removing the stacking location inside .bzr/branch. (Robert Collins, #376243)
- The log+ decorator, useful in debugging or profiling, could cause "AttributeError: 'list' object has no attribute 'next'". This is now fixed. The log decorator no longer shows the elapsed time or transfer rate because they're available in the log prefixes and the transport activity display respectively. (Martin Pool, #340347)
- Unshelve works correctly when multiple zero-length files are

- present on the shelf. (Aaron Bentley, #363444)
- Progress bars no longer show the network transport scheme or direction. (Martin Pool)
- launchpad-login now respects the 'verbose' option. (Jonathan Lange, #217031)

#### **Internals**

- bzrlib.user\_encoding is now officially deprecated. It is not possible to write a deprecation wrapper, but the variable will be removed in the near future. Use bzrlib.osutils.get\_user\_encoding() instead. (Alexander Belchenko)
- Command lookup has had hooks added. bzrlib.Command.hooks
  has three new hook points: get\_command, get\_missing\_command
  and list\_commands, which allow just-in-time command name
  provision rather than requiring all command names be known apriori. (Robert Collins)
- get\_app\_path from win32utils.py now supports REG\_EXPAND\_SZ data type and can read path to wordpad.exe. (Alexander Belchenko, #392046)
- graph.KnownGraph has been added. This is a class that can give answers to heads() very quickly. However, it has the assumption that the whole graph has already been loaded. This is true during annotate so it is used there with good success (as much as 2x faster for files with long ancestry and 'cherrypicked' changes.) (John Arbash Meinel, Vincent Ladeuil)
- OS file locks are now taken out using createFile rather than LockFileEx on Windows. The locking remains exclusive with LockFileEx but now it also works on older versions of Windows (such as Win98). (Martin <gzlist>)
- pack <=> pack fetching is now done via a PackStreamSource rather than the Packer code. The user visible change is that we now properly fetch the minimum number of texts for non-smart fetching. (John Arbash Meinel)
- VersionedFiles.\_add\_text is a new api that lets us insert text into the repository as a single string, rather than a list of lines.

This can improve memory overhead and performance of committing large files. (Currently a private api, used only by commit). (John Arbash Meinel)

#### **Improvements**

- bzr annotate can now be significantly faster. The time for bzr annotate NEWS is down to 7s from 22s in 1.16. Files with long histories and lots of 'duplicate insertions' will be improved more than others. (John Arbash Meinel, Vincent Ladeuil)
- bzr 1s is now faster. On OpenOffice.org, the time drops from 2.4 to 1.1 seconds. The improvement for bzr 1s -r-1 is more substantial dropping from 54.3 to 1.1 seconds. (lan Clatworthy)
- Improve "Path(s) are not versioned" error reporting for some commands. (Benoît PIERRE)
- Initial commit performance in --2a repositories has been improved by making it cheaper to build the initial CHKMap. (John Arbash Meinel)
- Resolving a revno to a revision id on a branch accessed via bzr:// or bzr+ssh:// is now much faster and involves no VFS operations. This speeds up commands like bzr pull -r 123. (Andrew Bennetts)
- revision-info now properly aligns the revnos/revids in the output and doesn't traceback when given revisions not in the current branch. Performance is also significantly improved when requesting multiple revs at once. (Matthew Fuller, John Arbash Meinel)
- Tildes are no longer escaped by Transports. (Andy Kilner)

# Documentation

- Avoid bad text wrapping in generated documentation. Slightly better formatting in the user reference. (Martin Pool, #249908)
- Minor clarifications to the help for End-Of-Line conversions. (Ian Clatworthy)

# **API Changes**

- Removed overspecific error class InvalidProgressBarType. (Martin Pool)
- The method ProgressView.\_show\_transport\_activity is now show\_transport\_activity because it's part of the contract between this class and the UI. (Martin Pool)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

### bzr 1.16.1

Released: 2009-06-26

End user testing of the 2a format revealed two serious bugs. The first, #365615, caused bzr to raise AbsentContentFactory errors when autopacking. This meant that commits or pushes to 2a-format repositories failed intermittently.

The second bug, #390563, caused the smart server to raise AbsentContentFactory when streaming 2a stacked 2a-format branches. This particularly affected branches stored on Launchpad in the 2a format.

Both of these bugs cause command failures only, neither of them cause data corruption or data loss. And, of course, both of these bugs are now fixed.

### **Bug Fixes**

- We now properly request a more minimal set of file texts when fetching multiple revisions. (Robert Collins, John Arbash Meinel, #390563)
- Repositories using CHK pages (which includes the new 2a format) will no longer error during commit or push operations when an autopack operation is triggered. (Robert Collins, #365615)
- chk\_map.iter\_interesting\_nodes now properly uses the intersection of referenced nodes rather than the union to determine what uninteresting pages we still need to look at. Prior to this, incrementally pushing to stacked branch would push the minimal data, but fetching everything would request extra texts. There are some unhandled cases wrt trees of different depths, but this fixes the common cases. (Robert Collins, John Arbash Meinel, #390563)
- GroupCompress repositories now take advantage of the pack hints parameter to permit cross-format fetching to incrementally pack the converted data. (Robert Collins)
- Repository.commit\_write\_group now returns opaque data about what was committed, for passing to the Repository.pack. Repositories without atomic commits will still return None. (Robert Collins)
- Repository.pack now takes an optional hint parameter which will support doing partial packs for repositories that can do that. (Robert Collins)
- RepositoryFormat has a new attribute 'pack\_compresses' which
  is True when doing a pack operation changes the compression
  of content in the repository. (Robert Collins)
- StreamSink and InterDifferingSerialiser will call Repository.pack with the hint returned by

Repository.commit\_write\_group if the formats were different and the repository can increase compression by doing a pack operation. (Robert Collins, #376748)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

# bzr 1.16

Codename: yesterday-in-california

**1.16rc1:** 2009-06-11 **1.16:** 2009-06-18

This version of Bazaar contains the beta release of the new 2a repository format, suitable for testing by fearless, advanced users. This format or an updated version of it will become the default format in Bazaar 2.0. Please read the NEWS entry before even thinking about upgrading to the new format.

Also included are speedups for many operations on huge projects, a bug fix for pushing stacked new stacked branches to smart servers and the usual bevy of bug fixes and improvements.

# Changes from 1.16rc1 to 1.16final

- Fix the nested tree flag check so that upgrade from development formats to 2a can work correctly. (Jelmer Vernooij, #388727)
- Automatic format upgrades triggered by default stacking policies on a 1.16rc1 (or later) smart server work again. (Andrew Bennetts, #388675)

# **Compatibility Breaks**

 Display prompt on stderr (instead of stdout) when querying users so that the output of commands can be safely redirected. (Vincent Ladeuil, #376582)

#### **New Features**

• A new repository format 2a has been added. This is a beta release of the brisbane-core (aka group-compress) project. This format now suitable for wider testing by advanced users willing to deal with some bugs. We would appreciate test reports, either positive or negative. Format 2a is substantially smaller and faster for many operations on many trees. This format or an updated version will become the default in bzr 2.0.

This is a rich-root format, so this repository format can be used with bzr-svn. Bazaar branches in previous non-rich-root formats can be converted (including by merge, push and pull) to format 2a, but not vice versa. We recommend upgrading previous development formats to 2a.

Upgrading to this format can take considerable time because it expands and more concisely repacks the full history.

If you use stacked branches, you must upgrade the stacked branches before the stacked-on branches. (See <a href="https://bugs.launchpad.net/bugs/374735">https://bugs.launchpad.net/bugs/374735</a>)

- --development7-rich-root is a new dev format, similar to --dev6 but using a Revision serializer using bencode rather than XML. (Jelmer Vernooij, John Arbash Meinel)
- mail\_client=claws now supports -body (and message body hooks). Also uses configured from address. (Barry Warsaw)

#### **Improvements**

- --development6-rich-root can now stack. (Modulo some smart-server bugs with stacking and non default formats.) (John Arbash Meinel, #373455)
- --development6-rich-root delays generating a delta index for the first object inserted into a group. This has a beneficial impact on bzr commit since each committed texts goes to its own group. For committing a 90MB file, it drops peak memory by about 200MB, and speeds up commit from 7s => 4s. (John Arbash Meinel)
- Numerous operations are now faster for huge projects, i.e. those
  with a large number of files and/or a large number of revisions,
  particularly when the latest development format is used. These
  operations (and improvements on OpenOffice.org) include:
  - branch in a shared repository (2X faster)
  - branch –no-tree (100X faster)
  - diff (2X faster)
  - tags (70X faster)

(Ian Clatworthy)

 Pyrex version of bencode support. This provides optimized support for both encoding and decoding, and is now found at bzrlib.bencode. bzrlib.utils.bencode is now deprecated. (Alexander Belchenko, Jelmer Vernooij, John Arbash Meinel)

#### **Bug Fixes**

- Bazaar can now pass attachment files to the mutt email client. (Edwin Grubbs, #384158)
- Better message in bzr add output suggesting using bzr ignored to see which files can also be added. (Jason Spashett, #76616)
- bzr pull -r 123 from a stacked branch on a smart server no longer fails. Also, the Branch.revision\_history() API now works in the same situation. (Andrew Bennetts, #380314)
- bzr serve on Windows no longer displays a traceback simply because a TCP client disconnected. (Andrew Bennetts)
- Clarify the rules for locking and fallback repositories. Fix bugs in how RemoteRepository was handling fallbacks along with the \_real\_repository. (Andrew Bennetts, John Arbash Meinel, #375496)
- Fix a small bug with fetching revisions w/ ghosts into a new stacked branch. Not often triggered, because it required ghosts to be part of the fetched revisions, not in the stacked-on ancestry. (John Arbash Meinel)
- Fix status and commit to work with content filtered trees, addressing numerous bad bugs with line-ending support. (Ian Clatworthy, #362030)
- Fix problem of "directory not empty" when contending for a lock over sftp. (Martin Pool, #340352)
- Fix rule handling so that eol is optional, not mandatory. (lan Clatworthy, #379370)
- Pushing a new stacked branch to a 1.15 smart server was broken due to a bug in the BzrDirFormat.initialize\_ex smart verb. This is fixed in 1.16, but required changes to the network protocol, so the BzrDirFormat.initialize\_ex verb has been removed and replaced with a corrected BzrDirFormat.initialize ex 1.16 verb. 1.15 clients will still

- work with a 1.16 server as they will fallback to slower (and bugfree) methods. (Jonathan Lange, Robert Collins, Andrew Bennetts, #385132)
- Reconcile can now deal with text revisions that originated in revisions that are ghosts. (Jelmer Vernooij, #336749)
- Support cloning of branches with ghosts in the left hand side history. (Jelmer Vernooij, #248540)
- The "bzr diff" now catches OSError from osutils.rmtree and logs a helpful message to the trace file, unless the temp directory really was removed (which would be very strange). Since the diff operation has succeeded from the user's perspective, no output is written to stderr or stdout. (Maritza Mendez, #363837)
- Translate errors received from a smart server in response to a BzrDirFormat.initialize Or BzrDirFormat.initialize\_ex request. This was causing tracebacks even for mundane errors like PermissionDenied. (Andrew Bennetts, #381329)

### Documentation

 Added directory structure and started translation of docs in Russian. (Alexey Shtokalo, Alexander Iljin, Alexander Belchenko, Dmitry Vasiliev, Volodymyr Kotulskyi)

#### **API Changes**

- Added osutils.parent\_directories(). (lan Clatworthy)
- bzrlib.progress.ProgressBar, ChildProgress, DotsProgressBar,
   TTYProgressBar and child\_progress are now deprecated; use
   ui\_factory.nested\_progress\_bar instead. (Martin Pool)
- graph.StackedParentsProvider is now a public API, replacing graph.\_StackedParentsProvider. The api is now considered stable and ready for external users. (Gary van der Merwe)
- bzrlib.user\_encoding is deprecated in favor of get\_user\_encoding. (Alexander Belchenko)
- TreeTransformBase no longer assumes that limbo is provided via disk. DiskTreeTransform now provides disk functionality. (Aaron Bentley)

## Internals

• Remove weave.py script for accessing internals of old weaveformat repositories. (Martin Pool)

### **Testing**

- make check no longer repeats the test run in LANG=C. (Martin Pool, #386180)
- The number of cores is now correctly detected on OSX. (John Szakmeister)
- The number of cores is also detected on Solaris and win32. (Vincent Ladeuil)
- The number of cores is also detected on FreeBSD. (Matthew Fuller)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

### bzr 1.15

**1.15rc1:** 2009-05-16 **1.15:** 2009-05-22 **1.15.1:** 2009-06-09

The smart server will no longer raise 'NoSuchRevision' when streaming content with a size mismatch in a reconstructed graph search. New command bzr dpush. Plugins can now define their own annotation tie-breaker when two revisions introduce the exact same line.

# Changes from 1.15.1 to 1.15.2

 Use zdll on Windows to build \_chk\_map\_pyx extension. (Alexander Belchenko)

# Changes from 1.15final to 1.15.1

Translate errors received from a smart server in response to a
 BzrDirFormat.initialize Or BzrDirFormat.initialize\_ex
 request. This was causing tracebacks even for mundane errors
 like PermissionDenied. (Andrew Bennetts, #381329)

# Changes from 1.15rc1 to 1.15final

• No changes

# **Compatibility Breaks**

bzr 1s is no longer recursive by default. To recurse, use the new -R option. The old --non-recursive option has been removed. If you alias 1s to 1s -R, you can disable recursion using --no-recursive instead. (lan Clatworthy)

#### **New Features**

- New command bzr dpush that can push changes to foreign branches (svn, git) without setting custom bzr-specific metadata. (Jelmer Vernooij)
- The new development format --development6-rich-root now supports stacking. We chose not to use a new format marker, since old clients will just fail to open stacked branches, the same as if we used a new format flag. (John Arbash Meinel, #373455)
- Plugins can now define their own annotation tie-breaker when two revisions introduce the exact same line. See bzrlib.annotate.\_break\_annotation\_tie Be aware though that this is temporary, private (as indicated by the leading '\_') and a first step to address the problem. (Vincent Ladeuil, #348459)
- New command bzr dpush that can push changes to foreign branches (svn, git) without setting custom bzr-specific metadata. (Jelmer Vernooij)
- bzr send will now check the <a href="mailto:child\_submit\_format">child\_submit\_format</a> setting in the submit branch to determine what format to use, if none was specified on the command-line. (Jelmer Vernooij)

### **Improvements**

- -Dhpss output now includes the number of VFS calls made to the remote server. (Jonathan Lange)
- --coverage works for code running in threads too. (Andrew Bennets, Vincent Ladeuil)
- bzr pull now has a --local option to only make changes to the local branch, and not the bound master branch. (Gary van der Merwe, #194716)
- bzr rm \* is now as fast as bzr rm \* --keep. (Johan Walles, #180116)

#### **Bug Fixes**

- Adding now works properly when path contains a symbolic link. (Geoff Bache, #183831)
- An error is now raised for unknown eol values. (Brian de Alwis, #358199)
- bzr merge --weave will now generate a conflict if one side deletes a line, and the other side modifies the line. (John Arbash Meinel, #328171)
- bzr reconfigure --standalone no longer raises
   IncompatibleRepositories. (Martin von Gagern, #248932)
- bzr send works to send emails again using MAPI. (Neil Martinsen-Burrell, #346998)
- Check for missing parent inventories in StreamSink. This
  prevents incomplete stacked branches being created by 1.13
  bzr:// and bzr+ssh:// clients (which have bug #354036). Instead,
  the server now causes those clients to send the missing
  records. (Andrew Bennetts)
- Correctly handle http servers proposing multiple authentication schemes. (Vincent Ladeuil, #366107)
- End-Of-Line content filters are now loaded correctly. (lan Clatworthy, Brian de Alwis, #355280)
- Fix a bug in the pure-python GroupCompress code when handling copies longer than 64KiB. (John Arbash Meinel, #364900)
- Fix TypeError in running bzr break-lock on some URLs. (Alexander Belchenko, Martin Pool, #365891)
- Non-recursive bzr 1s now works properly when a path is specified. (Jelmer Vernooij, #357863)
- ssh usernames (defined in ~/.ssh/config) are honoured for bzr+ssh connections. (Vincent Ladeuil, #367726)
- Several bugs related to unicode symlinks have been fixed and the test suite enhanced to better catch regressions for them.

#### (Vincent Ladeuil)

- The smart server will no longer raise 'NoSuchRevision' when streaming content with a size mismatch in a reconstructed graph search: it assumes that the client will make sure it is happy with what it got, and this sort of mismatch is normal for stacked environments. bzr 1.13.0/1 will stream from unstacked branches only in that case not getting all the content expected would be a bug. However the graph search is how we figured out what we wanted, so a mismatch is both odd and unrecoverable without starting over, and starting over will end up with the same data as if we just permitted the mismatch. If data is gc'd, doing a new search will find only the truncated data, so sending only the truncated data seems reasonable. bzr versions newer than this will stream from stacked branches and check the stream to find missing content in the stacked-on branch, and thus will handle the situation implicitly. (Robert Collins, #360791)
- Upgrading to, or fetching into a 'rich-root' format will now correctly set the root data the same way that reconcile does. (Robert Collins, #368921)
- Using unicode Windows API to obtain command-line arguments. (Alexander Belchenko, #375934)

# Documentation

### **API Changes**

- InterPackRepo.fetch and RepoFetcher now raise NoSuchRevision instead of InstallFailed when they detect a missing revision. InstallFailed itself has been deleted. (Jonathan Lange)
- Not passing arguments to bzrlib.commands.main() will now grab
  the arguments from osutils.get\_unicode\_argv() which has
  proper support for unicode arguments on windows. Further, the
  supplied arguments are now required to be unicode strings,
  rather than user encoded strings. (Alexander Belchenko)

#### **Internals**

- bzrlib.branch.Branch.set\_parent is now present on the base branch class and will call \_set\_parent\_location after doing unicode encoding. (Robert Collins)
- bzrlib.remote.RemoteBranch.\_set\_parent\_location will use a new verb Branch.set\_parent\_location removing further VFS operations. (Robert Collins)
- bzrlib.bzrdir.BzrDir.\_get\_config now returns a
   TransportConfig or similar when the dir supports configuration
   settings. The base class defaults to None. There is a matching
   new server verb BzrDir.get-config\_file to reduce roundtrips
   for getting BzrDir configuration. (Robert Collins)
- bzrlib.tests.ExtendedTestResult has new methods startTests called before the first test is started, done called after the last test completes, and a new parameter strict. (Robert Collins)
- -Dhpss when passed to bzr will cause a backtrace to be printed when VFS operations are started on a smart server repository. This should not occur on regular push and pull operations, and is a key indicator for performance regressions. (Robert Collins)
- -Dlock when passed to the selftest (e.g. bzr -Dlock selftest) will cause mismatched physical locks to cause test errors rather than just reporting to the screen. (Robert Collins)
- -Dprogress will cause pdb to start up if a progress view jumps backwards. (Robert Collins)
- Fallback CredentialStore instances registered with fallback=True are now be able to provide credentials if obtaining credentials via ~/.bazaar/authentication.conf fails. (Jelmer Vernooij, Vincent Ladeuil, #321918)
- New hook Lock.lock\_broken which runs when a lock is broken. This is mainly for testing that lock/unlock are balanced in tests.

(Vincent Ladeuil)

- New MergeDirective hook 'merge\_request\_body' allows hooks to supply or alter a body for the message produced by bzr\_send.
- New smart server verb <a href="mailto:BzrDir.initialize\_ex">BzrDir.initialize\_ex</a> which implements a refactoring to the core of clone allowing less round trips on new branches. (Robert Collins)
- New method <a href="mailto:rename\_revisions">Tags.rename\_revisions</a> that can rename revision ids tags are pointing at. (Jelmer Vernooij)
- Updated the bundled configobj library to 4.6.0 (Matt Nordhoff)

# **Testing**

 bzr selftest will now fail if lock/unlock are not correctly balanced in tests. Using -Dlock will turn the related failures into warnings. (Vincent Ladeuil, Robert Collins)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

### bzr 1.14

Codename:brisbane-core1.14rc1:2009-04-061.14rc2:2009-04-191.14:2009-04-281.14.1:2009-05-01

New formats 1.14 and 1.14-rich-root supporting End-Of-Line (EOL) conversions, keyword templating (via the bzr-keywords plugin) and generic content filtering. End-of-line conversion is now supported for formats supporting content filtering.

# Changes from 1.14final to 1.14.1

 Change api\_minimum\_version back to api\_minimum\_version = (1, 13, 0)

# Changes from 1.14rc2 to 1.14final

• Fix a bug in the pure-python groupcompress code when handling copies longer than 64KiB. (John Arbash Meinel, #364900)

#### Changes from 1.14rc1 to 1.14rc2

- Fix for bug 358037 Revision not in bzrlib.groupcompress.GroupCompressVersionedFiles (Brian de Alwis, John A Meinel)
- Fix for bug 354036 ErrorFromSmartServer -AbsentContentFactory object has no attribute 'get\_bytes\_as' exception while pulling from Launchpad (Jean-Francois Roy, Andrew Bennetts, Robert Collins)
- Fix for bug 355280 eol content filters are never loaded and thus never applied (Brian de Alwis, Ian Clatworthy)
- bzr.dev -r4280 Change \_fetch\_uses\_deltas = False for CHK repos until we can write a better fix. (John Arbash Meinel, Robert Collins)
- Fix for bug 361574 uncommit recommends undefined —levels and -n options (Marius Kruger, Ian Clatworthy)
- bzr.dev r4289 as cherrypicked at lp:~spiv/bzr/stackingcherrypick-1.14 (Andrew Bennetts, Robert Collins)

#### **Compatibility Breaks**

- A previously disabled code path to accelerate getting configuration settings from a smart server has been reinstated.
   We think this *may* cause a incompatibility with servers older than bzr 0.15. We intend to issue a point release to address this if it turns out to be a problem. (Robert Collins, Andrew Bennetts)
- bzr no longer autodetects nested trees as 'tree-references'.
   They must now be explicitly added tree references. At the commandline, use join –reference instead of add. (Aaron Bentley)
- The --long log format (the default) no longer shows merged revisions implicitly, making it consistent with the short and line log formats. To see merged revisions for just a given revision, use bzr log -no -rx. To see every merged revision, use bzr log -no. (lan Clatworthy)

#### **New Features**

- New formats 1.14 and 1.14-rich-root supporting End-Of-Line (EOL) conversions, keyword templating (via the bzr-keywords plugin) and generic content filtering. These formats replace the experimental development-wt5 and development-wt5-rich-root formats respectively, but have support for filtered views disabled. (Ian Clatworthy)
- New mv --auto option recognizes renames after they occur.
   (Aaron Bentley)
- bzr can now get passwords from stdin without requiring a controlling terminal (i.e. by redirecting stdin). (Vincent Ladeuil)
- bzr log now supports filtering of multiple files and directories and will show changes that touch any of them. Furthermore, directory filtering now shows the changes to any children of that directory, not just the directory object itself. (lan Clatworthy, #97715)
- bzr shelve can now apply changes without storing anything on the shelf, via the new –destroy option. (Aaron Bentley)
- bzr send now accepts –body to specify an initial message body.
   (Aaron bentley)
- bzr xxx --usage where xxx is a command now shows a usage message and the options without the descriptive help sections (like Description and Examples). A message is also given explaining how to see the complete help, i.e. bzr help xxx. (lan Clatworthy)
- Content filters can now be used to provide custom conversion between the canonical format of content (i.e. as stored) and the convenience format of content (i.e. as created in working trees).
   See bzr help content-filters for further details. (lan Clatworthy, Alexander Belchenko)
- End-of-line conversion is now supported for formats supporting

content filtering. See bzr help eol for details. (lan Clatworthy)

• Newly-blessed *join* command allows combining two trees into one. (Aaron Bentley)

#### **Improvements**

- A new format name alias default-rich-root has been added and points at the closest relative of the default format that supports rich roots. (Jelmer Vernooij, #338061)
- Branching from a stacked branch using bzr\*:// will now stream the data when the target repository does not need topological ordering, reducing round trips and network overhead. This uses the existing smart server methods added in 1.13, so will work on any 1.13 or newer server. (Robert Collins, Andrew Bennetts)
- bzr cat and bzr export now supports a --filters option that displays/saves the content after content filters are applied. (Ian Clatworthy)
- bzr ignore gives a more informative message when existing version controlled files match the ignore pattern. (Neil Martinsen-Burrell, #248895)
- bzr log now has --include-merges as an alias for --levels 0.
   (lan Clatworthy)
- bzr send is faster on repositories with deep histories. (lan Clatworthy)
- IPv6 literals are accepted in URLs. (stlman, Martin Pool, Jelmer Vernooij, #165014)
- Progress bars now show the rate of network activity for bzr+ssh:// and bzr:// connections. (Andrew Bennetts)
- Prompt for user names if they are not in the configuration. (Jelmer Vernooij, #256612)
- Pushing to a stacked pack-format branch on a 1.12 or older smart server now takes many less round trips. (Andrew Bennetts, Robert Collins, #294479)
- Streaming push can be done to older repository formats. This is implemented using a new Repository.insert\_stream\_locked RPC. (Andrew Bennetts, Robert Collins)

- The "ignoring files outside view: .." message has been reworded to "Ignoring files outside view. View is .." to reduce confusion about what was being considered and what was being ignored. (Ian Clatworthy)
- The long log formatter now shows [merge] indicators. If only one level of revisions is displayed and merges are found, the long and short log formatters now tell the user how to see the hidden merged revisions. (lan Clatworthy)
- The brisbane-core project has delivered its beta format development6-rich-root. This format is suitable for judicious testing by early adopters. In particular if you are benchmarking bzr performance please be sure to test using this format. At this stage more information is best obtained by contacting the Bazaar mailing list or IRC channel if you are interested in using this format. We will make end user documentation available closer to blessing the format as production ready. (Robert Collins, John Arbash Meinel, Ian Clatworthy, Vincent Ladeuil, Andrew Bennetts, Martin Pool)
- Tildes are no longer escaped. No more %7Euser/project/branch! (Jonathan Lange)

#### **Bug Fixes**

- Pushing a new stacked branch will also push the parent inventories for revisions at the stacking boundary. This makes sure that the stacked branch has enough data to calculate inventory deltas for all of its revisions (without requiring the fallback branch). This avoids "AbsentContentFactory' object has no attribute 'get\_bytes\_as'" errors when fetching the stacked branch from a 1.13 (or later) smart server. This partially fixes #354036. (Andrew Bennetts, Robert Collins)
- End-Of-Line content filters are now loaded correctly. (lan Clatworthy, Brian de Alwis, #355280)
- Authentication plugins now receive all the parameters from the request itself (aka host, port, realm, path, etc). Previously, only the authentication section name, username and encoded password were provided. (Jean-Francois Roy)
- bzr gives a better message if an invalid regexp is passed to bzr
   log -m. (Anne Mohsen, Martin Pool)
- bzr split now says "See also: join" (Aaron Bentley, #335015)
- bzr version-info now works in empty branches. (Jelmer Vernooij, #313028)
- Fix "is not a stackable format" error when pushing a stackableformat branch with an unstackable-format repository to a destination with a default stacking policy. (Andrew Bennetts)
- Fixed incorrect "Source format does not support stacking" warning when pushing to a smart server. (Andrew Bennetts, #334114)
- Fix 'make check-dist-tarball' failure by converting paths to unicode when needed. (Vincent Ladeuil, #355454)
- Fixed "Specified file 'x/y/z' is outside current view: " occurring on bzr add x/y/z in formats supporting views when no view is defined. (Ian Clatworthy, #344708)

- It is no longer possible to fetch between repositories while the target repository is in a write group. This prevents race conditions that prevent the use of RPC's to perform fetch, and thus allows optimising more operations. (Robert Collins, Andrew Bennetts)
- merge --force works again. (Robert Collins, #342105)
- No more warnings are issued about sha being deprecated under python-2.6. (Vincent Ladeuil, #346593)
- Pushing a new branch to a server that has a stacking policy will now upgrade from the local branch format when the stacking policy points at a branch which is itself stackable, because we know the client can read both branches, we know that the trunk for the project can be read too, so the upgrade will not inconvenience users. (Robert Collins, #345169)
- Pushing a new stacked branch will also push the parent inventories for revisions at the stacking boundary. This makes sure that the stacked branch has enough data to calculate inventory deltas for all of its revisions (without requiring the fallback branch). This avoids "AbsentContentFactory' object has no attribute 'get\_bytes\_as'" errors when fetching the stacked branch from a 1.13 (or later) smart server. This partially fixes #354036. (Andrew Bennetts, Robert Collins)
- The full test suite is passing again on OSX. Several minor issues (mostly test related) have been fixed. (Vincent Ladeuil, #355273).
- The GNU Changelog formatter is slightly improved in the case where the delta is empty, and now correctly claims not to support tags. (Andrea Bolognani)
- Shelve can now shelve changes to a symlink target. (James Westby, #341558)
- The help for the info command has been corrected. (lan Clatworthy, #351931)
- Upgrade will now use a sensible default format if the source repository uses rich roots. (Jelmer Vernooij, #252908)

#### Documentation

- Expanded the index of the developer documentation. (Eric Siegerman)
- New topic bzr help debug-flags. (Martin Pool)
- The generated manpage now explicitly lists aliases as commands. (James Westby, #336998)

#### **API Changes**

- APIs deprecated in 1.6 and previous versions of bzr are now removed. (Martin Pool)
- CommitReporter is no longer called with unchanged status during commit - this was a full-tree overhead that bzr no longer performs. (Robert Collins)
- New abstract UIFactory method get\_username which will be called to obtain the username to use when connecting to remote machines. (Jelmer Vernooij)
- New API Inventory.filter() added that filters an inventory by a set of file-ids so that only those fileids, their parents and their children are included. (Ian Clatworthy)
- New sort order for get\_record\_stream groupcompress which sorts optimally for use with groupcompress compressors. (John Arbash Meinel, Robert Collins)
- Repository APIs get\_deltas\_for\_revisions() and get\_revision\_delta() now support an optional specific\_fileids parameter. If provided, the deltas are filtered so that only those file-ids, their parents and their children are included. (Ian Clatworthy)
- The get\_credentials and set\_credentials methods of AuthenticationConfig now accept an optional realm argument. (Jean-Francois Roy)
- The pb argument to fetch() is deprecated. (Martin Pool)
- The serializer class and the serializer format registry have moved from bzrlib.xml\_serializer to bzrlib.serializer. (Jelmer Vernooij)
- The smart server jail now hooks into BzrDir.open to prevent any BzrDir that is not inside the backing transport from being opened. See the module documentation for

- bzrlib.smart.request for details. (Andrew Bennetts, Robert Collins)
- Tree.get\_symlink\_target now always returns a unicode string result or None. Previously it would return the bytes from reading the link which could be in any arbitrary encoding. (Robert Collins)

## **Testing**

- bzrlib.tests.TestCase now fails the test if its own setup and tearDown weren't called. This catches faulty tests that forget to upcall when overriding setup and tearDown. Those faulty tests were not properly isolated. (Andrew Bennetts, Robert Collins)
- Fix test\_msgeditor.MsgEditorTest test isolation. (Vincent Ladeuil, #347130)
- medusa is not used anymore as an FTP test server starting with python2.6. A new FTP test server based on pyftplib can be used instead. This new server is a soft dependency as medusa which is still preferred if both are available (modulo python version). (Vincent Ladeuil)

#### Internals

- Added chk\_map for fast, trie-based storage of tuple to string maps. (Robert Collins, John Arbash Meinel, Vincent Ladeuil)
- Added bzrlib.chk\_map for fast, trie-based storage of tuple to string maps. (Robert Collins, John Arbash Meinel, Vincent Ladeuil)
- Added bzrlib.inventory\_delta module. This will be used for serializing and deserializing inventory deltas for more efficient streaming on the network. (Robert Collins, Andrew Bennetts)
- Branch.\_get\_config has been added, which splits out access to the specific config file from the branch. This is used to let RemoteBranch avoid constructing real branch objects to access configuration settings. (Robert Collins, Andrew Bennetts)
- Branch now implements set\_stacked\_on\_url in the base class as the implementation is generic and should impact foreign formats. This helps performance for RemoteBranch push operations to new stacked branches. (Robert Collins, Andrew Bennetts)
- BtreeIndex.\_spill\_mem\_keys\_to\_disk() now generates disk index with optmizations turned off. This only has effect when processing > 100,000 keys during something like bzr pack. (John Arbash Meinel)
- bzr selftest now accepts --subunit to run in subunit output mode. Requires lp:subunit installed to work, but is not a hard dependency. (Robert Collins)
- BzrDir.open\_branch now takes an optional ignore\_fallbacks parameter for controlling opening of stacked branches. (Andrew Bennetts, Robert Collins)
- CommitBuilder has a new method, record\_iter\_changes which works in terms of an iter\_changes iterator rather than full tree

- scanning. (Robert Collins)
- DirState can now be passed a custom SHA1Provider Object enabling it to store the SHA1 and stat of the canonical (post content filtered) form. (Ian Clatworthy)
- New assertLength method based on one Martin has squirreled away somewhere. (Robert Collins, Martin Pool)
- New hook BzrDir.pre\_open which runs before opening BzrDir objects, allowing better enforcement of the smart server jail when dealing with stacked branches. (Robert Collins, Andrew Bennetts)
- New hook RioversionInfoBuilder.revision, allowing extra entries to be added to the stanza that is printed for a particular revision. (Jelmer Vernooij)
- New repository method refresh\_data to cause any repository to make visible data inserted into the repository by a smart server fetch operation. (Robert Collins, Andrew Bennetts)
- register\_filter\_stack\_map now takes an optional fallback parameter, a callable to invoke if a preference has a value not in the map of filter stacks. This enhancement allows, for example, bzr-svn to handle existing svn properties that define a list of keywords to be expanded. (Ian Clatworthy)
- RemoteBranchConfig will use a new verb Branch.set\_config\_option to write config settings to smart servers that support this, saving 5 round trips on the stacked streaming acceptance test. (Robert Collins, Andrew Bennetts)
- RemoteBranch now provides \_get\_config for access to just the branch specific configuration from a remote server, which uses the already existing Branch.get\_config\_file smart verb. (Robert Collins, Andrew Bennetts)
- RemoteRepository will now negatively cache missing revisions during get\_parent\_map while read-locked. Write-locks are unaffected. (Robert Collins, Andrew Bennetts)
- Removed InterRemoteToOther, InterOtherToRemote and

- InterPackToRemotePack classes, as they are now unnecessary. (Andrew Bennetts)
- RepositoryFormat as a new attribute fast\_deltas to indicate whether the repository can efficiently generate deltas between trees regardless of tree size. (Robert Collins)
- Repository.iter\_files\_bytes() now properly returns an "iterable of byte strings" (aka 'chunked') for the content. It previously was returning a plain string, which worked, but performed very poorly when building a working tree (file.writelines(str) is very inefficient). This can have a large effect on bzr checkout times. (John Arbash Meinel)
- selftest now supports a –parallel option, with values of 'fork' or 'subprocess' to run the test suite in parallel. Currently only linux machine work, other platforms need patches submitted. (Robert Collins, Vincent Ladeuil)
- tests.run\_suite has a new parameter suite\_decorators, a list of callables to use to decorate the test suite. Such decorators can add or remove tests, or even remote the test suite to another machine if desired. (Robert Collins)
- The smart server verb Repository.get\_parent\_map can now include information about ghosts when the special revision include-missing: is in the requested parents map list. With this flag, ghosts are included as missing:REVISION\_ID. (Robert Collins, Andrew Bennetts)
- \_walk\_to\_common\_revisions will now batch up at least 50 revisions before calling get\_parent\_map on the target, regardless of InterRepository. (Andrew Bennetts, Robert Collins)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

## bzr 1.13

Codename: paraskavedekatriaphobia

1.13:2009-03-141.13rc1:2009-03-101.13.1:2009-03-231.13.2:2009-04-27

GNU Changelog output can now be produced by bzr log --gnu-changelog. Debug flags can now be set in ~/.bazaar/bazaar.conf. Lightweight checkouts and stacked branches should both be much faster over remote connections.

## Changes From 1.13.1 to 1.13.2

A regression was found in the 1.13.1 release. When bzr 1.13.1 and earlier push a stacked branch they do not take care to push all the parent inventories for the transferred revisions. This means that a smart server serving that branch often cannot calculate inventory deltas for the branch (because smart server does not/cannot open fallback repositories). Prior to 1.13 the server did not have a verb to stream revisions out of a repository, so that's why this bug has appeared now.

# **Bug Fixes**

 Fix for bug 354036 ErrorFromSmartServer -AbsentContentFactory object has no attribute 'get\_bytes\_as' exception while pulling from Launchpad (Jean-Francois Roy, Andrew Bennetts, Robert Collins)

# Changes From 1.13final to 1.13.1

A couple regessions where found in the 1.13 release. The pyrexgenerated C extensions are missing from the .tar.gz and .zip files. Documentation on how to generate GNU ChangeLogs is wrong.

## **Bug Fixes**

- Change ./bzr's \_script\_version to match ./bzrlib/\_\_init\_\_.py version\_info. (Bob Tanner, Martin Pool, #345232)
- Distribution archives for 1.13 do not contain generated C extension modules (Jean-Francois Roy, Bob Tanner, #344465)
- GNU ChangeLog output can now be produced by bzr log format gnu-changelog is incorrect (Deejay, Bob Tanner, Martin Pool, Robert Collins, #343928)
- merge --force works again. (Robert Collins, #342105)

## Changes From 1.13rc1 to 1.13final

- Fix "is not a stackable format" error when pushing a stackableformat branch with an unstackable-format repository to a destination with a default stacking policy. (Andrew Bennetts)
- Progress bars now show the rate of network activity for bzr+ssh:// and bzr:// connections. (Andrew Bennetts)

# **Compatibility Breaks**

• bzr log --line now indicates which revisions are merges with [merge] after the date. Scripts which parse the output of this command may need to be adjusted. (Neil Martinsen-Burrell)

#### **New Features**

- bzr reconfigure now supports –with-trees and –with-no-trees options to change the default tree-creation policy of shared repositories. (Matthew Fuller, Marius Kruger, #145033)
- Debug flags can now be set in ~/.bazaar/bazaar.conf. (Martin Pool)
- Filtered views provide a mask over the tree so that users can focus on a subset of a tree when doing their work. See Filtered views in chapter 7 of the User Guide and bzr help view for details. (lan Clatworthy)
- GNU Changelog output can now be produced by bzr log -- gnu-changelog. (Andrea Bolognani, Martin Pool)
- The -Dmemory flag now gives memory information on Windows. (John Arbash Meinel)
- Multiple authors for a commit can now be recorded by using the "–author" option multiple times. (James Westby, #185772)
- New clean-tree command, from bzrtools. (Aaron Bentley, Jelmer Vernooij)
- New command bzr launchpad-open opens a Launchpad web page for that branch in your web browser, as long as the branch is on Launchpad at all. (Jonathan Lange)
- New API for getting bugs fixed by a revision: Revision.iter bugs(). (Jonathan Lange)

### **Improvements**

- All bzr classes registered are now in Hooks bzrlib.hooks.known\_hooks. This removes the separate list from bzrlib.tests and ensures that all hooks registered there are correctly isolated by the test suite (previously MutableTreeHooks were not being isolated correctly). Further, documentation for hooks is now dynamically generated from the present HookPoints. bzr hooks will now also report on all the hooks present in the bzrlib.hooks.known hooks registry. (Robert Collins)
- bzr add no longer prints add completed on success. Failure still prints an error message. (Robert Collins)
- bzr branch now has a --no-tree option which turns off the generation of a working tree in the new branch. (Daniel Watkins, John Klinger, #273993)
- Bazaar will now point out bzr+ssh:// to the user when they use ssh://. (Jelmer Vernooij, #330535)
- bzr -v info now omits the number of committers branch statistic, making it many times faster for large projects. To include that statistic in the output, use bzr -vv info. (lan Clatworthy)
- bzr push to a bzr url (bzr://, bzr+ssh:// etc) will stream if the server is version 1.13 or greater, reducing roundtrips significantly. (Andrew Bennetts, Robert Collins)
- Lightweight Checkouts and Stacked Branches should both be much faster over remote connections. Building the working tree now batches up requests into approx 5MB requests, rather than a separate request for each file. (John Arbash Meinel)
- Support for GSSAPI authentication when using HTTP or HTTPS. (Jelmer Vernooij)

- The bzr shelve prompt now includes a '?' help option to explain the short options better. (Daniel Watkins, #327429)
- bzr 1p-open now falls back to the push location if it cannot find a public location. (Jonathan Lange, #332372)
- bzr lp-open will try to find the Launchpad URL for the location passed on the command line. This makes bzr lp-open lp:foo work as expected. (Jonathan Lange, #332705)
- bzr send now supports MH-E via emacsclient. (Eric Gillespie)

### **Bug Fixes**

- Allows bzr log <FILE> to be called in an empty branch without backtracing. (Vincent Ladeuil, #346431)
- Bazaar now gives a better message including the filename if it's unable to read a file in the working directory, for example because of a permission error. (Martin Pool, #338653)
- bzr cat -r<old> <path> doesn't traceback anymore when <path> has a file id in the working tree different from the one in revision <old>. (Vincent Ladeuil, #341517, #253806)
- bzr send help is more specific about how to apply merge directives. (Neil Martinsen-Burrell, #253470)
- bzr missing now uses Repository.get\_revision\_delta() rather than fetching trees and determining a delta itself. (Jelmer Vernooij, #315048)
- bzr push to a smart server no longer causes "Revision {set([('null:',)])} not present ..." errors when the branch has multiple root revisions. (Andrew Bennetts, #317654)
- bzr shelve now properly handle patches with no terminating newline. (Benoît PIERRE, #303569)
- bzr unshelve gives a more palatable error if passed a noninteger shelf id. (Daniel Watkins)
- Export now handles files that are not present in the tree. (James Westby, #174539)
- Fixed incorrect "Source format does not support stacking" warning when pushing to a smart server. (Andrew Bennetts, #334114)
- Fixed "sprout() got an unexpected keyword argument 'source\_branch'" error branching from old repositories. (Martin Pool, #321695)
- Make bzr push --quiet <non-local location> less chatty. (Kent Gibson, #221461)

- Many Branch hooks would not fire with bzr:// and bzr+ssh:// branches, and this was not noticed due to a bug in the test logic for branches. This is now fixed and a test added to prevent it reoccuring. (Robert Collins, Andrew Bennetts)
- Restore the progress bar on Windows. We were disabling it when TERM wasn't set, but Windows doesn't set TERM. (Alexander Belchenko, #334808)
- setup.py build\_ext now gives a proper error when an extension fails to build. (John Arbash Meinel)
- Symlinks to non ascii file names are now supported. (Robert Collins, Vincent Ladeuil, #339055, #272444)
- Under rare circumstances (aka nobody reported a bug about it), the ftp transport could revert to ascii mode. It now stays in binary mode except when needed. (Vincent Ladeuil)
- Unshelve does not generate warnings about progress bars. (Aaron Bentley, #328148)
- shelve cleans up properly when unversioned files are specified.
   (Benoît Pierre, Aaron Bentley)

### Documentation

- Added organizing your workspace to the User Guide appendices, summarizing some common ways of organizing trees, branches and repositories and the processes/workflows implied/enabled by each. (lan Clatworthy)
- Hooks can now be self documenting.
   bzrlib.hooks.Hooks.create\_hook is the entry point for this feature. (Robert Collins)
- The documentation for shelve and unshelve has been clarified. (Daniel Watkins, #327421, #327425)

## **API Changes**

- bzr selftest now fails if the bazaar sources contain trailing whitespace, non-unix style line endings and files not ending in a newline. About 372 files and 3243 lines with trailing whitespace was updated to comply with this. The code already complied with the other criteria, but now it is enforced. (Marius Kruger)
- bzrlib.branch.PushResult was renamed to
   bzrlib.branch.BranchPushResult. (Jelmer Vernooij)
- Branch.fetch and Repository.fetch now return None rather than a count of copied revisions and failed revisions. A while back we stopped ever reporting failed revisions because we started erroring instead, and the copied revisions count is not used in the UI at all indeed it only reflects the repository status not changes to the branch itself. (Robert Collins)
- Inventory.apply\_delta now raises an AssertionError if a file-id appears multiple times within the delta. (Ian Clatworthy)
- MutableTree.commit now favours the "authors" argument, with the old "author" argument being deprecated.
- Remove deprecated EmptyTree. (Martin Pool)
- Repository.fetch now accepts an optional fetch\_spec parameter. A SearchResult or MiniSearchResult may be passed to fetch\_spec instead of a last\_revision to specify exactly which revisions to fetch. (Andrew Bennetts)
- RepositoryAcquisitionPolicy.acquire\_repository now returns a tuple of (repository, is\_new\_flag), rather than just the repository. (Andrew Bennetts)
- Revision.get\_apparent\_author() is now deprecated, replaced by Revision.get\_apparent\_authors(), which returns a list. The former now returns the first item that would be returned from the second.
- The BranchBuilder test helper now accepts a timestamp

- parameter to build\_commit and build\_snapshot. (Martin Pool)
- The \_fetch\_\* attributes on Repository are now on RepositoryFormat, more accurately reflecting their intent (they describe a disk format capability, not state of a particular repository of that format). (Robert Collins)

#### Internals

- Branching from a non-stacked branch on a smart protocol is now free of virtual file system methods. (Robert Collins, Andrew Bennetts)
- Branch and Repository creation on a bzr+ssh://server are now done via RPC calls rather than VFS calls, reducing round trips for pushing new branches substantially. (Robert Collins)
- Branch.clone now takes the repository\_policy formerly used inside BzrDir.clone\_on\_transport, allowing stacking to be configured before the branch tags and revision tip are set. This fixes a race condition cloning stacked branches that would cause plugins to have hooks called on non-stacked instances. (Robert Collins, #334187)
- BzrDir.cloning\_metadir now has a RPC call. (Robert Collins)
- BzrDirFormat.\_\_str\_\_ now uses the human readable description rather than the sometimes-absent disk label. (Robert Collins)
- bzrlib.fetch is now composed of a sender and a sink component allowing for decoupling over a network connection. Fetching from or into a RemoteRepository with a 1.13 server will use this to stream the operation. (Andrew Bennetts, Robert Collins)
- bzrlib.tests.run\_suite accepts a runner\_class parameter supporting the use of different runners. (Robert Collins)
- Change how file\_ids and revision\_ids are interned as part of inventory deserialization. Now we use the real intern(), rather than our own workaround that would also cache a Unicode copy of the string, and never emptied the cache. This should slightly reduce memory consumption. (John Arbash Meinel)
- New branch method create\_clone\_on\_transport that returns a branch object. (Robert Collins)
- New hook Commands['extend\_command'] to allow plugins to

- access a command object before the command is run (or help generated from it), without overriding the command. (Robert Collins)
- New version of the BzrDir.find\_repository verb supporting \_network\_name to support removing more \_ensure\_real calls. (Robert Collins)
- RemoteBranchFormat no longer claims to have a disk format string. (Robert Collins)
- Repository objects now have suspend\_write\_group and resume\_write\_group methods. These are currently only useful with pack repositories. (Andrew Bennetts, Robert Collins)
- BzrDirFormat, BranchFormat and RepositoryFormat objects now have a network\_name for passing the format across RPC calls. (Robert Collins, Andrew Bennetts)
- RepositoryFormat objects now all have a new attribute
   \_serializer used by fetch when reserialising is required.
   (Robert Collins, Andrew Bennetts)
- Some methods have been pulled up from BzrBranch to Branch to aid branch types that are not bzr branch objects (like RemoteBranch). (Robert Collins, Andrew Bennetts)
- Test adaptation has been made consistent throughout the built in tests. TestScenarioApplier, multiply\_tests\_from\_modules, adapt\_tests, adapt\_modules have all been deleted. Please use multiply\_tests, or for lower level needs apply\_scenarios and apply\_scenario. (Robert Collins)
- TestSkipped is now detected by TestCase and passed to the TestResult by calling addskip. For older TestResult objects, where addskip is not available, addError is still called. This permits test filtering in subunit to strip out skipped tests resulting in a faster fix-shrink-list-run cycle. This is compatible with the testtools protocol for skips. (Robert Collins)
- The \_index of KnitVersionedFiles now supports the ability to

scan an underlying index that is going to be incorporated into the KnitVersionedFiles object, to determine if it has missing delta references. The method is <a href="mailto:scan\_unvalidated\_index">scan\_unvalidated\_index</a>. (Andrew Bennetts, Robert Collins)

- There is a RemoteSink object which handles pushing to smart servers. (Andrew Bennetts, Robert Collins)
- TransportTraceDecorator now logs put\_bytes\_non\_atomic and rmdir calls. (Robert Collins)
- VersionedFiles record adapters have had their signature change from (record, record.get\_bytes\_as(record.storage\_kind)) to (record) reducing excess duplication and allowing adapters to access private data in record to obtain content more efficiently. (Robert Collins)
- We no longer probe to see if we should create a working tree during clone if we cannot get a local\_abspath for the new bzrdir. (Robert Collins)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

## bzr 1.12

**Codename:** 1234567890 **1.12:** 2009-02-13 **1.12rc1:** 2009-02-10

This release of Bazaar contains many improvements to the speed, documentation and functionality of bzr log and the display of logged revisions by bzr status. bzr now also gives a better indication of progress, both in the way operations are drawn onto a text terminal, and by showing the rate of network IO.

# Changes from RC1 to Final

- bzr init --development-wt5[-rich-root] would fail because of circular import errors. (John Arbash Meinel, #328135)
- Expanded the help for log and added a new help topic called log-formats. (lan Clatworthy)

## **Compatibility Breaks**

- By default, bzr status after a merge now shows just the pending merge tip revisions. This improves the signal-to-noise ratio after merging from trunk and completes much faster. To see all merged revisions, use the new -v flag. (Ian Clatworthy)
- bzr log --line now shows any tags after the date and before the commit message. If you have scripts which parse the output from this command, you may need to adjust them accordingly. (lan Clatworthy)
- bzr log --short now shows any additional revision properties after the date and before the commit message. Scripts that parse output of the log command in this situation may need to adjust. (Neil Martinsen-Burrell)
- The experimental formats 1.12-preview and 1.12-preview-rich-root have been renamed development-wt5 and development-wt5-rich-root respectively, given they are not ready for release in 1.12. (lan Clatworthy)
- read\_bundle\_from\_url has been deprecated. (Vincent Ladeuil)

#### **New Features**

- Add support for filtering bzr missing on revisions. Remote revisions can be filtered using bzr missing -r -20..-10 and local revisions can be filtered using bzr missing --my-revision -20..-10. (Marius Kruger)
- bzr log -p displays the patch diff for each revision. When logging a file, the diff only includes changes to that file. (Ian Clatworthy, #202331, #227335)
- bzr log supports a new option called -n N or --level N. A value of 0 (zero) means "show all nested merge revisions" while a value of 1 (one) means "show just the top level". Values above 1 can be used to see a limited amount of nesting. That can be useful for seeing the level or two below PQM submits for example. To force the --short and --line formats to display all nested merge revisions just like --long does by default, use a command like bzr log --short -n0. To display just the mainline using --long format, bzr log --long -n1. (lan Clatworthy)

### **Improvements**

- bzr add more clearly communicates success vs failure. (Daniel Watkins)
- bzr init will now print a little less verbose output. (Marius Kruger)
- bzr log is now much faster in many use cases, particularly at incrementally displaying results and filtering by a revision range. (lan Clatworthy)
- bzr log --short and bzr log --line now show tags, if any, for each revision. The tags are shown comma-separated inside {}.
   For short format, the tags appear at the end of line before the optional [merge] indicator. For line format, the tags appear after the date. (lan Clatworthy)
- Progress bars now show the rate of activity for some sftp operations, and they are drawn different. (Martin Pool, #172741)
- Progress bars now show the rate of activity for urllib and pycurl based http client implementations. The operations are tracked at the socket level for better precision. (Vincent Ladeuil)
- Rule-based preferences can now accept multiple patterns for a set of rules. (Marius Kruger)
- The ancestor: revision spec will now default to referring to the parent of the branch if no other location is given. (Daniel Watkins, #198417)
- The debugger started as a result of setting \$BZR\_PDB works around a bug in pdb, http://bugs.python.org/issue4150. The bug can cause truncated tracebacks in Python versions before 2.6. (Andrew Bennetts)
- VirtualVersionedFiles now implements iter\_lines\_added\_or\_present\_in\_keys. This allows the creation of new branches based on stacked bzr-svn branches. (#311997)

### **Bug Fixes**

- bzr annotate --show-ids doesn't give a backtrace on empty files anymore. (Anne Mohsen, Vincent Ladeuil, #314525)
- bzr log FILE now correctly shows mainline revisions merging a change to FILE when the --short and --line log formats are used. (lan Clatworthy, #317417)
- bzr log -rx..Y FILE now shows the history of FILE provided it existed in Y or X, even if the file has since been deleted or renamed. If no range is given, the current/basis tree and initial tree are searched in that order. More generally, log now interprets filenames in their historical context. (Ian Clatworthy, #175520)
- bzr status now reports nonexistent files and continues, then errors (with code 3) at the end. (Karl Fogel, #306394)
- Don't require the present compression base in knits to be the same when adding records in knits. (Jelmer Vernooij, #307394)
- Fix a problem with CIFS client/server lag on Windows colliding with an invariant-per-process algorithm for generating AtomicFile names (Adrian Wilkins, #304023)
- Many socket operations now handle EINTR by retrying the operation. Previously EINTR was treated as an unrecoverable failure. There is a new until\_no\_eintr helper function in bzrlib.osutils. (Andrew Bennetts)
- Support symlinks with non-ascii characters in the symlink filename. (Jelmer Vernooij, #319323)
- There was a bug in how we handled resolving when a file is deleted in one branch, and modified in the other. If there was a criss-cross merge, we would cause the deletion to conflict a second time. (Vincent Ladeuil, John Arbash Meinel)
- There was another bug in how we chose the correct intermediate LCA in criss-cross merges leading to several kind

- of changes be incorrectly handled. (John Arbash Meinel, Vincent Ladeuil)
- Unshelve now handles deleted paths without crashing. (Robert Collins)

# Documentation

• Improved plugin developer documentation. (Martin Pool)

## **API Changes**

- ProgressBarStack is deprecated; instead use ui\_factory.nested\_progress\_bar to create new progress bars. (Martin Pool)
- ForeignVcsMapping() now requires a ForeignVcs object as first argument. (Jelmer Vernooij)
- ForeignVcsMapping.show\_foreign\_revid() has been moved to ForeignVcs. (Jelmer Vernooij)
- read\_bundle\_from\_url is deprecated in favor of read\_mergeable\_from\_url. (Vincent Ladeuil)
- Revision specifiers are now registered in bzrlib.revisionspec.revspec\_registry, and the old list of revisionspec classes (bzrlib.revisionspec.SPEC\_TYPES) has been deprecated. (Jelmer Vernooij, #321183)
- The progress and UI classes have changed; the main APIs remain the same but code that provides a new UI or progress bar class may need to be updated. (Martin Pool)

#### **Internals**

- Default User Interface (UI) is CLIUIFactory when bzr runs in a dumb terminal. It is sometimes desirable do override this default by forcing bzr to use TextUIFactory. This can be achieved by setting the BZR\_USE\_TEXT\_UI environment variable (emacs shells, as opposed to compile buffers, are such an example). (Vincent Ladeuil)
- New API Branch.iter\_merge\_sorted\_revisions() that iterates
   over (revision\_id, depth, revno, end\_of\_merge) tuples. (lan
   Clatworthy)
- New Branch.dotted\_revno\_to\_revision\_id() and Branch.revision\_id\_to\_dotted\_revno() APIs that pick the most efficient way of doing the mapping. (Ian Clatworthy)
- Refactor cmd\_serve so that it's a little easier to build commands that extend it, and perhaps even a bit easier to read. (Jonathan Lange)
- TreeDelta.show() now accepts a filter parameter allowing log formatters to retrict the output. (Vincent Ladeuil)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

## bzr 1.11

Codename: "Eyes up!" Released: 2009-01-19

This first monthly release of Bazaar for 2009 improves Bazaar's operation in Windows, Mac OS X, and other situations where file names are matched without regard to capitalization: Bazaar tries to match the case of an existing file. This release of Bazaar also improves the efficiency of Tortoise Windows Shell integration and lets it work on 64-bit platforms.

The UI through which Bazaar supports historic formats has been improved, so 'bzr help formats' now gives a simpler and shorter list, with clear advice.

This release also fixes a number of bugs, particularly a glitch that can occur when there are concurrent writes to a pack repository.

## **Bug Fixes**

- Fix failing test when CompiledChunksToLines is not available. (Vincent Ladeuil)
- Stacked branches don't repeatedly open their transport connection. (John Arbash Meinel)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

## bzr 1.11rc1

Codename: "Eyes up!"
Released: 2009-01-09

## Changes

• Formats using Knit-based repository formats are now explicitly marked as deprecated. (Ian Clatworthy)

#### **New Features**

- Add support for *bzr tags -r 1..2*, that is we now support showing tags applicable for a specified revision range. (Marius Kruger)
- authentication.conf now accepts pluggable read-only credential stores. Such a plugin (netrc\_credential\_store) is now included, handles the \$HOME/.netrc file and can server as an example to implement other plugins. (Vincent Ladeuil)
- shelve --list can now be used to list shelved changes. (Aaron Bentley)

#### **Improvements**

- Add trailing slash to directories in all output of bzr 1s, except
   bzr 1s --null. (Gordon P. Hemsley, #306424)
- bzr revision-info now supports a -d option to specify an alternative branch. (Michael Hudson)
- Add connection to a C++ implementation of the Windows Shell Extension which is able to fully replace the current Python implemented one. Advantages include 64bit support and reduction in overhead for processes which drag in shell extensions. (Mark Hammond)
- Support the Claws mail client directly, rather than via xdg-email.
   This prevents the display of an unnecessary modal dialog in Claws, informing the user that a file has been attached to the message, and works around bug #291847 in xdg-utils which corrupts the destination address.
- When working on a case-insensitive case-preserving filesystem, as commonly found with Windows, bzr will often ignore the case of the arguments specified by the user in preference to the case of an existing item on the file-system or in the inventory to help prevent counter-intuitive behaviour on Windows. (Mark Hammond)

#### **Bug Fixes**

- Allow BzrDir implementation to implement backing up of control directory. (#139691)
- bzr push creating a new stacked branch will now only open a single connection to the target machine. (John Arbash Meinel)
- Don't call iteritems on transport\_list\_registry, because it may change during iteration. (Martin Pool, #277048)
- Don't make a broken branch when pushing an unstackableformat branch that's in a stackable shared repository to a location with default stack-on location. (Andrew Bennetts, #291046)
- Don't require embedding user in HTTP(S) URLs do use authentication.conf. (Ben Jansen, Vincent Ladeuil, #300347)
- Fix a problem with CIFS client/server lag on windows colliding with an invariant-per-process algorithm for generating AtomicFile names (Adrian Wilkins, #304023)
- Fix bogus setUp signature in UnavailableFTPServer. (Gary van der Merwe, #313498)
- Fix compilation error in \_dirstate\_helpers\_c on SunOS/Solaris. (Jari Aalto)
- Fix SystemError in \_patiencediff\_c module by calling PyErr\_NoMemory() before returning NULL in PatienceSequenceMatcher new. (Andrew Bennetts, #303206)
- Give proper error message for diff with non-existent dotted revno. (Marius Kruger, #301969)
- Handle EACCES (permission denied) errors when launching a message editor, and emit warnings when a configured editor cannot be started. (Andrew Bennetts)
- \$HOME/.netrc file is now recognized as a read-only credential store if configured in authentication.conf with 'password\_encoding=netrc' in the appropriate sections. (Vincent

- Ladeuil, #103029)
- Opening a stacked branch now properly shares the connection, rather than opening a new connection for the stacked-on branch. (John Arbash meinel)
- Preserve transport decorators while following redirections.
   (Vincent Ladeuil, #245964, #270863)
- Provides a finer and more robust filter for accepted redirections.
   (Vincent Ladeuil, #303959, #265070)
- shelve paths are now interpreted relative to the current working tree. (Aaron Bentley)
- Transport.readv() defaults to not reading more than 100MB in a single array. Further RemoteTransport.readv sets this to 5MB to work better with how it splits its requests. (John Arbash Meinel, #303538)
- Pack repositories are now able to reload the pack listing and retry the current operation if another action causes the data to be repacked. (John Arbash Meinel, #153786)
- pull -v now respects the log\_format configuration variable.
   (Aaron Bentley)
- push -v now works on non-initial pushes. (Aaron Bentley)
- Use the short status format when the short format is used for log. (Vincent Ladeuil, #87179)
- Allow files to be renamed or moved via remove + add-by-id. (Charles Duffy, #314251)

## Documentation

• Improved the formats help topic to explain why multiple formats exist and to provide guidelines in selecting one. Introduced two new supporting help topics: current-formats and other-formats. (lan Clatworthy)

## **API Changes**

- LRUCache(after\_cleanup\_size) was renamed to after\_cleanup\_count and the old name deprecated. The new name is used for clarity, and to avoid confusion with LRUSizeCache(after\_cleanup\_size). (John Arbash Meinel)
- New ForeignRepository base class, to help with foreign branch support (e.g. svn). (Jelmer Vernooij)
- node\_distances and select\_farthest can no longer be imported from bzrlib.graph. They can still be imported from bzrlib.deprecated\_graph, which has been the preferred way to import them since before 1.0. (Andrew Bennetts)
- The logic in commit now delegates inventory basis calculations to the commitBuilder object; this requires that the commit builder in use has been updated to support the new recording\_deletes and record\_delete methods. (Robert Collins)

### **Testing**

- An HTTPS server is now available (it requires python-2.6). Future bzr versions will allow the use of the python-2.6 ssl module that can be installed for 2.5 and 2.4.
- bzr selftest now fails if new trailing white space is added to the bazaar sources. It only checks changes not committed yet. This means that PQM will now reject changes that introduce new trailing whitespace. (Marius Kruger)
- Introduced new experimental formats called 1.12-preview and 1.12-preview-rich-root to enable testing of related pending features, namely content filtering and filtered views. (lan Clatworthy)

#### **Internals**

- Added an InventoryEntry cache when deserializing inventories.
   Can cut the time to iterate over multiple RevisionsTrees in half.
   (John Arbash Meinel)
- Added bzrlib.fifo\_cache.FIFoCache which is designed to have minimal overhead versus using a plain dict for cache hits, at the cost of not preserving the 'active' set as well as an LRUCache. (John Arbash Meinel)
- bzrlib.patience\_diff.unified\_diff now properly uses a tab character to separate the filename from the date stamp, and doesn't add trailing whitespace when a date stamp is not supplied. (Adeodato Simó, John Arbash Meinel)
- DirStateWorkingTree and DirStateWorkingTreeFormat added as base classes of WorkingTree4 and WorkingTreeFormat4 respectively. (Ian Clatworthy)
- KnitVersionedFiles.\_check\_should\_delta() now uses the get\_build\_details api to avoid multiple hits to the index, and to properly follow the compression\_parent rather than assuming it is the left-hand parent. (John Arbash Meinel)
- KnitVersionedFiles.get\_record\_stream() will now chose a more optimal ordering when the keys are requested 'unordered'. Previously the order was fully random, now the records should be returned from each pack in turn, in forward I/O order. (John Arbash Meinel)
- mutter() will now flush the ~/.bzr.log if it has been more than
   2s since the last time it flushed. (John Arbash Meinel)
- New method bzrlib.repository.Repository.add\_inventory\_by\_delta allows adding an inventory via an inventory delta, which can be more efficient for some repository types. (Robert Collins)

- Repository CommitBuilder objects can now accumulate an inventory delta. To enable this functionality call builder.recording\_deletes and additionally call builder.record\_delete when a delete against the basis occurs. (Robert Collins)
- The default http handler has been changed from pycurl to urllib. The default is still pycurl for https connections. (The only advantage of pycurl is that it checks ssl certificates.) (John Arbash Meinel)
- VersionedFiles.get\_record\_stream() can now return objects with a storage\_kind of chunked. This is a collection (list/tuple) of strings. You can use osutils.chunks\_to\_lines() to turn them into guaranteed 'lines' or you can use ''.join(chunks) to turn it into a fulltext. This allows for some very good memory savings when asking for many texts that share ancestry, as the individual chunks can be shared between versions of the file. (John Arbash Meinel)
- pull -v and push -v use new function bzrlib.log.show\_branch\_change (Aaron Bentley)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

## bzr 1.10

Released: 2008-12-05

Bazaar 1.10 has several performance improvements for copying revisions (especially for small updates to large projects). There has also been a significant amount of effort in polishing stacked branches. The commands shelve and unshelve have become core commands, with an improved implementation.

The only changes versus bzr-1.10rc1 are bugfixes for stacked branches.

## bug Fixes

- Don't set a pack write cache size from RepoFetcher, because the cache is not coherent with reads and causes ShortReadvErrors. This reverses the change that fixed #294479. (Martin Pool, #303856)
- Properly handle when a revision can be inserted as a delta versus when it needs to be expanded to a fulltext for stacked branches. There was a bug involving merge revisions. As a method to help prevent future difficulties, also make stacked fetches sort topologically. (John Arbash Meinel, #304841)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

## bzr 1.10rc1

**Released:** 2008-11-28

This release of Bazaar focuses on performance improvements when pushing and pulling revisions, both locally and to remote networks. The popular shelve and unshelve commands, used to interactively revert and restore work in progress, have been merged from bzrtools into the bzr core. There are also bug fixes for portability, and for stacked branches.

#### **New Features**

- New commit\_message\_template hook that is called by the commit code to generate a template commit message. (Jelmer Vernooij)
- New *shelve* and *unshelve* commands allow undoing and redoing changes. (Aaron Bentley)

#### **Improvements**

- (Remote)Branch.copy\_content\_into no longer generates the full revision history just to set the last revision info. (Andrew Bennetts, John Arbash Meinel)
- Fetches between formats with different serializers (such as pack-0.92-subtree and 1.9-rich-root) are faster now. This is due to operating on batches of 100 revisions at time rather than one-by-one. (Andrew Bennetts, John Arbash Meinel)
- Search index files corresponding to pack files we've already used before searching others, because they are more likely to have the keys we're looking for. This reduces the number of iix and tix files accessed when pushing 1 new revision, for instance. (John Arbash Meinel)
- Signatures to transfer are calculated more efficiently in item\_keys\_introduced\_by. (Andrew Bennetts, John Arbash Meinel)
- The generic fetch code can once again copy revisions and signatures without extracting them completely to fulltexts and then serializing them back down into byte strings. This is a significant performance improvement when fetching from a stacked branch. (John Arbash Meinel, #300289)
- When making a large readv() request over bzr+ssh, break up the request into more manageable chunks. Because the RPC is not yet able to stream, this helps keep us from buffering too much information at once. (John Arbash Meinel)

#### **Bug Fixes**

- Better message when the user needs to set their Launchpad ID. (Martin Pool, #289148)
- bzr commit --local doesn't access the master branch anymore. This fixes a regression introduced in 1.9. (Marius Kruger, #299313)
- Don't call the system <a href="chdir">chdir</a>() with an empty path. Sun OS seems to give an error in that case. Also, don't count on <a href="getcwd">getcwd</a>() being able to allocate a new buffer, which is a gnu extension. (John Arbash Meinel, Martin Pool, Harry Hirsch, #297831)
- Don't crash when requesting log –forward <file> for a revision range starting with a dotted revno. (Vincent Ladeuil, #300055)
- Don't create text deltas spanning stacked repositories; this could cause "Revision X not present in Y" when later accessing them. (Martin Pool, #288751)
- Pack repositories are now able to reload the pack listing and retry the current operation if another action causes the data to be repacked. (John Arbash Meinel, #153786)
- PermissionDenied errors from smart servers no longer cause "PermissionDenied: "None"" on the client. (Andrew Bennetts, #299254)
- Pushing to a stacked pack repository now batches writes, the same way writes are batched to ordinary pack repository. This makes pushing to a stacked branch over the network much faster. (Andrew Bennetts, #294479)
- TooManyConcurrentRequests no longer occur when a fetch fails and tries to abort a write group. This allows the root cause (e.g. a network interruption) to be reported. (Andrew Bennetts, #297014)
- RemoteRepository.get\_parent\_map now uses fallback

repositories. (Aaron Bentley, #297991?, #293679?)

## **API Changes**

- CommitBuilder now validates the strings it will be committing, to ensure that they do not have characters that will not be properly round-tripped. For now, it just checks for characters that are invalid in the XML form. (John Arbash Meinel, #295161)
- Constructor parameters for NewPack (internal to pack repositories) have changed incompatibly.
- Repository.abort\_write\_group now accepts an optional suppress\_errors flag. Repository implementations that override abort\_write\_group will need to be updated to accept the new argument. Subclasses that only override \_abort\_write\_group don't need to change.
- Transport implementations must provide copy\_tree\_to\_transport. A default implementation is provided for Transport subclasses.

## **Testing**

- bzr selftest now fails if no doctests are found in a module that's expected to have them. (Martin Pool)
- Doctests now only report the first failure. (Martin Pool)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

## bzr 1.9

**Released:** 2008-11-07

This release of Bazaar adds a new repository format, 1.9, with smaller and more efficient index files. This format can be specified when creating a new repository, or used to losslessly upgrade an existing repository. bzr 1.9 also speeds most operations over the smart server protocol, makes annotate faster, and uses less memory when making checkouts or pulling large amounts of data.

## **Bug Fixes**

- Fix "invalid property value 'branch-nick' for None" regression with branches bound to svn branches. (Martin Pool, #293440)
- Fix SSL/https on Python2.6. (Vincent Ladeuil, #293054)
- SFTPTransport.readv() had a bug when requests were out-oforder. This only triggers some-of-the-time on Knit format repositories. (John Arbash Meinel, #293746)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

# bzr 1.9rc1

**Released:** 2008-10-31

#### **New Features**

- New Branch hook transform\_fallback\_location allows a function to be called when looking up the stacked source. (Michael Hudson)
- New repository formats 1.9 and 1.9-rich-root. These have all the functionality of 1.6, but use the new btree indexes. These indexes are both smaller and faster for access to historical information. (John Arbash Meinel)

#### **Improvements**

- BTreeIndex code now is able to prefetch extra pages to help tune the tradeoff between bandwidth and latency. Should be tuned appropriately to not impact commands which need minimal information, but provide a significant boost to ones that need more context. Only has a direct impact on the --development2 format which uses btree's for the indexes. (John Arbash Meinel)
- bzr dump-btree is a hidden command introduced to allow dumping the contents of a compressed btree file. (John Arbash Meinel)
- bzr pack now tells the index builders to optimize for size. For btree index repositories, this can save 25% of the index size (mostly in the text indexes). (John Arbash Meinel)
- bzr push to an existing branch or repository on a smart server is faster, due to Bazaar making more use of the get\_parent\_map RPC when querying the remote branch's revision graph. (Andrew Bennetts)
- default username for bzr+ssh and sftp can be configured in authentication.conf. (Aaron Bentley)
- launchpad-login now provides a default username for bzr+ssh and sftp URLs, allowing username-free URLs to work for everyone. (Aaron Bentley)
- 1p: lookups no longer include usernames, making them shareable and shorter. (Aaron Bentley)
- New PackRepository.autopack smart server RPC, which does autopacking entirely on the server. This is much faster than autopacking via plain file methods, which downloads a large amount of pack data and then re-uploads the same pack data into a single file. This fixes a major (although infrequent) cause of lengthy delays when using a smart server. For example,

pushing the 10th revision to a repository with 9 packs now takes 44 RPCs rather than 179, and much less bandwidth too. This requires Bazaar 1.9 on both the client and the server, otherwise the client will fallback to the slower method. (Andrew Bennetts)

#### **Bug Fixes**

- A failure to load a plugin due to an IncompatibleAPI exception is now correctly reported. (Robert Collins, #279451)
- API versioning support now has a multiple-version checking api require\_any\_api. (Robert Collins, #279447)
- bzr branch --stacked from a smart server to a standalone branch works again. This fixes a regression in 1.7 and 1.8. (Andrew Bennetts, #270397)
- bzr co uses less memory. It used to unpack the entire WT into memory before writing it to disk. This was a little bit faster, but consumed lots of memory. (John Arbash Meinel, #269456)
- bzr missing --quiet no longer prints messages about whether there are missing revisions. The exit code indicates whether there were or not. (Martin Pool, #284748)
- Fixes to the annotate code. The fast-path which re-used the stored deltas was accidentally disabled all the time, instead of only when a branch was stacked. Second, the code would accidentally re-use a delta even if it wasn't against the leftparent, this could only happen if bzr reconcile decided that the parent ordering was incorrect in the file graph. (John Arbash Meinel)
- "Permission denied" errors that occur when pushing a new branch to a smart server no longer cause tracebacks. (Andrew Bennetts, #278673)
- Some compatibility fixes for building the extensions with MSVC and for python2.4. (John Arbash Meinel, #277484)
- The index logic is now able to reload the list of pack files if and index ends up disappearing. We still don't reload if the pack data itself goes missing after checking the index. This bug appears as a transient failure (file not found) when another process is writing to the repository. (John Arbash Meinel,

#153786)

 bzr switch and bzr bind will now update the branch nickname if it was previously set. All checkouts will now refer to the bound branch for a nickname if one was not explicitly set. (Marius Kruger, #230903)

## Documentation

• Improved hook documentation. (Michael Ernst)

# **API Changes**

• commands.plugins\_cmds is now a CommandRegistry, not a dict.

## Internals

• New AuthenticationConfig.set\_credentials method allows easy programmatic configuration of authetication credentials.

## bzr 1.8

Released: 2008-10-16

Bazaar 1.8 includes several fixes that improve working tree performance, display of revision logs, and merges. The bzr testsuite now passes on OS X and Python 2.6, and almost completely passes on Windows. The smartserver code has gained several bug fixes and performance improvements, and can now run server-side hooks within an http server.

## **Bug Fixes**

• Fix "Must end write group" error when another error occurs during bzr push. (Andrew Bennetts, #230902)

## **Portability**

• Some Pyrex versions require the WIN32 macro defined to compile on that platform. (Alexander Belchenko, Martin Pool, #277481)

# bzr 1.8rc1

**Released:** 2008-10-07

## Changes

- bzr log file has been changed. It now uses a different method for determining which revisions to show as merging the changes to the file. It now only shows revisions which merged the change towards your mainline. This simplifies the output, makes it faster, and reduces memory consumption. (John Arbash Meinel)
- bzr merge now defaults to having --reprocess set, whenever -- show-base is not supplied. (John Arbash Meinel)
- bzr+http// will now optionally load plugins and write logs on the server. (Marius Kruger)
- bzrlib.\_dirstate\_helpers\_c.pyx does not compile correctly with Pyrex 0.9.4.1 (it generates C code which causes segfaults). We explicitly blacklist that version of the compiler for that extension. Packaged versions will include .c files created with pyrex >= 0.9.6 so it doesn't effect releases, only users running from the source tree. (John Arbash Meinel, #276868)

### **Features**

• bzr is now compatible with python-2.6. python-2.6 is not yet officially supported (nor released, tests were conducted with the dev version of python-2.6rc2), but all known problems have been fixed. Feedback welcome. (Vincent Ladeuil, #269535)

#### **Improvements**

- bzr annotate will now include uncommitted changes from the local working tree by default. Such uncommitted changes are given the revision number they would get if a commit was done, followed with a ? to indicate that its not actually known. (Robert Collins, #3439)
- bzr branch now accepts a --standalone option, which creates a standalone branch regardless of the presence of shared repositories. (Daniel Watkins)
- bzr push is faster in the case there are no new revisions to push. It is also faster if there are no tags in the local branch. (Andrew Bennetts)
- File changes during a commit will update the tree stat cache.
   (Robert Collins)
- Location aliases can now accept a trailing path. (Micheal Hudson)
- New hooks Lock.hooks when LockDirs are acquired and released. (Robert Collins, MartinPool)
- Switching in heavyweight checkouts uses the master branch's context, not the checkout's context. (Adrian Wilkins)
- status on large trees is now faster, due to optimisations in the walkdirs code. Of particular note, the walkdirs code now performs a temporary <a href="chdir">chdir</a>() while reading a single directory; if your platform has non thread-local current working directories (and is not windows which has its own implementation), this may introduce a race condition during concurrent uses of bzrlib. The bzrlib CLI will not encounter this as it is single threaded for working tree operations. (Robert Collins)
- The C extensions now build on python 2.4 (Robert Collins, #271939)
- The -Dhpss debug flag now reports the number of smart server

calls per medium to stderr. This is in addition to the existing detailed logging to the .bzr.log trace file. (Andrew Bennetts)

#### **Bug Fixes**

- Avoid random failures arising from misinterpreted errno values in \_readdir\_pyx.read\_dir. (Martin Pool, #279381)
- Branching from a shared repository on a smart server into a new repository now preserves the repository format. (Andrew Bennetts, #269214)
- bzr log now accepts a --change option. (Vincent Ladeuil, #248427)
- bzr missing now accepts an --include-merges option. (Vincent Ladeuil, #233817)
- Don't try to filter (internally) '.bzr' from the files to be deleted if it's not there. (Vincent Ladeuil, #272648)
- Fix '\_in\_buffer' AttributeError when using the -Dhpss debug flag. (Andrew Bennetts)
- Fix TooManyConcurrentRequests errors caused by a connection failure when doing bzr pull or bzr merge from a bzr+ssh URL. (Andrew Bennetts, #246233)
- Fixed bzr st -r branch: PATH\_TO\_BRANCH where the other branch is in a different repository than the current one. (Lukáš Lalinský, #144421)
- Make the first line of the manpage preamble a comment again.
   (David Futcher, #242106)
- Remove use of optional parameter in GSSAPI FTP support, since it breaks newer versions of Python-Kerberos. (Jelmer Vernooij)
- The autopacking logic will now always create a single new pack from all of the content which it deems is worth moving. This avoids the 'repack a single pack' bug and should result in better packing overall. (John Arbash Meinel, #242510, #172644)
- Trivial documentation fix. (John Arbash Meinel, #270471)
- bzr switch and bzr bind will now update the branch nickname if

it was previously set. All checkouts will now refer to the bound branch for a nickname if one was not explicitly set. (Marius Kruger, #230903)

## Documentation

• Explain revision/range identifiers. (Daniel Clemente)

### **API Changes**

- CommitBuilder.record\_entry\_contents returns one more element in its result tuple an optional file system hash for the hash cache to use. (Robert Collins)
- dirstate.DirState.update\_entry will now only calculate the sha1 of a file if it is likely to be needed in determining the output of iter\_changes. (Robert Collins)
- The PackRepository, RepositoryPackCollection, NewPack classes have a slightly changed interface to support different index types; as a result other users of these classes need to supply the index types they want. (Robert Collins)

### **Testing**

- bzrlib.tests.repository\_implementations has been renamed to bzrlib.tests.per\_repository so that we have a common structure (and it is shorter). (John Arbash Meinel, #239343)
- LocalTransport.abspath() now returns a drive letter if the transport has one, fixing numerous tests on Windows. (Mark Hammond)
- PreviewTree is now tested via intertree\_implementations.
   (Aaron Bentley)
- The full test suite is passing again on OSX. (Guillermo Gonzalez, Vincent Ladeuil)
- The full test suite passes when run with -Eallow\_debug. (Andrew Bennetts)

#### **Internals**

- A new hook, Branch.open, has been added, which is called when branch objects are opened. (Robert Collins)
- bzrlib.osutils.\_walkdirs\_utf8 has been refactored into common tree walking, and modular directory listing code to aid future performance optimisations and refactoring. (Robert Collins)
- bzrlib.trace.debug\_memory can be used to get a quick memory dump in the middle of processing. It only reports memory if /proc/PID/status is available. (John Arbash Meinel)
- New method RevisionSpec.as\_tree for representing the revision specifier as a revision tree object. (Lukáš Lalinský)
- New race-free method on MutableTree get\_file\_with\_stat for use when generating stat cache results. (Robert Collins)
- New win32utils.get\_local\_appdata\_location() provides access to a local directory for storing data. (Mark Hammond)
- To be compatible with python-2.6 a few new rules should be observed. 'message' attribute can't be used anymore in exception classes, 'sha' and 'md5' modules have been deprecated (use osutils.[md5|sha]), object\_\_init\_\_ and object.\_\_new\_\_ don't accept parameters anymore. (Vincent Ladeuil)

## bzr 1.7.1

**Released:** 2008-10-01

No changes from 1.7.1rc1.

## bzr 1.7.1rc1

**Released:** 2008-09-24

This release just includes an update to how the merge algorithm handles file paths when we encounter complex history.

#### **Features**

 If we encounter a criss-cross in history, use information from direct Least Common Ancestors to resolve inventory shape (locations of files, adds, deletes, etc). This is similar in concept to using --lca for merging file texts, only applied to paths. (John Arbash Meinel)

### bzr 1.7

Released: 2008-09-23

This release includes many bug fixes and a few performance and feature improvements. bzr rm will now scan for missing files and remove them, like how bzr add scans for unknown files and adds them. A bit more polish has been applied to the stacking code. The b-tree indexing code has been brought in, with an eye on using it in a future repository format. There are only minor installer changes since bzr-1.7rc2.

#### **Features**

• Some small updates to the win32 installer. Include localization files found in plugins, and include the builtin distutils as part of packaging qbzr. (Mark Hammond)

## bzr 1.7rc2

Released: 2008-09-17

A few bug fixes from 1.7rc1. The biggest change is a new RemoteBranch.get\_stacked\_on\_url rpc. This allows clients that are trying to access a Stacked branch over the smart protocol, to properly connect to the stacked-on location.

### **Bug Fixes**

- Branching from a shared repository on a smart server into a new repository now preserves the repository format. (Andrew Bennetts, #269214)
- Branching from a stacked branch via bzr+ssh can properly connect to the stacked-on branch. (Martin Pool, #261315)
- bzr init no longer re-opens the BzrDir multiple times. (Vincent Ladeuil)
- Fix '\_in\_buffer' AttributeError when using the -Dhpss debug flag. (Andrew Bennetts)

## bzr 1.7rc1

Released: 2008-09-09

This release candidate for bzr 1.7 has several bug fixes and a few performance and feature improvements. bzr rm will now scan for missing files and remove them, like how bzr add scans for unknown files and adds them. A bit more polish has been applied to the stacking code. The b-tree indexing code has been brought in, with an eye on using it in a future repository format.

## Changes

- bzr export can now export a subdirectory of a project. (Robert Collins)
- bzr remove-tree will now refuse to remove a tree with uncommitted changes, unless the --force option is specified. (Lukáš Lalinský, #74101)
- bzr rm will now scan for files that are missing and remove just them automatically, much as bzr add scans for new files that are not ignored and adds them automatically. (Robert Collins)

### **Features**

- Support for GSSAPI authentication when using FTP as documented in RFC2228. (Jelmer Vernooij, #49623)
- Add support for IPv6 in the smart server. (Jelmer Vernooij, #165014)

#### **Improvements**

- A url like log+file://tmp will log all access to that Transport to .bzr.log, which may help in debugging or profiling. (Martin Pool)
- bzr branch and bzr push use the default stacking policy if the branch format supports it. (Aaron Bentley)
- bzr init and bzr init-repo will now print out the same as bzr info if it completed successfully. (Marius Kruger)
- bzr uncommit logs the old tip revision id, and displays how to restore the branch to that tip using bzr pull. This allows you to recover if you realize you uncommitted the wrong thing. (John Arbash Meinel)
- Fix problems in accessing stacked repositories over bzr://.
   (Martin Pool, #261315)
- SFTPTransport.readv() was accidentally using list += string, which 'works', but adds each character separately to the list, rather than using list.append(string). Fixing this makes the SFTP transport a little bit faster (~20%) and use a bit less memory. (John Arbash Meinel)
- When reading index files, if we happen to read the whole file in a single request treat it as a \_buffer\_all request. This happens most often on small indexes over remote transports, where we default to reading 64kB. It saves a round trip for each small index during fetch operations. Also, if we have read more than 50% of an index file, trigger a \_buffer\_all on the next request. This works around some inefficiencies because reads don't fall neatly on page boundaries, so we would ignore those bytes, but request them again later. This could trigger a total read size of more than the whole file. (John Arbash Meinel)

#### **Bug Fixes**

- bzr rm is now aliased to bzr del for the convenience of svn users. (Robert Collins, #205416)
- Catch the infamous "select/poll returned error" which occurs when pycurl try to send a body request to an HTTP/1.0 server which has already refused to handle the request. (Vincent Ladeuil, #225020)
- Fix ObjectNotLocked errors when using various commands (including bzr cat and bzr annotate) in combination with a smart server URL. (Andrew Bennetts, #237067)
- FTPTransport.stat() would return 0000 as the permission bits for the containing .bzr/ directory (it does not implement permissions). This would cause us to set all subdirectories to 0700 and files to 0600 rather than leaving them unmodified. Now we ignore 0000 as the permissions and assume they are invalid. (John Arbash Meinel, #259855)
- Merging from a previously joined branch will no longer cause a traceback. (Jelmer Vernooij, #203376)
- Pack operations on windows network shares will work even with large files. (Robert Collins, #255656)
- Running bzr st PATH\_TO\_TREE will no longer suppress merge status. Status is also about 7% faster on mozilla sized trees when the path to the root of the tree has been given. Users of the internal show\_tree\_status function should be aware that the show\_pending flag is now authoritative for showing pending merges, as it was originally. (Robert Collins, #225204)
- Set valid default \_param\_name for Option so that ListOption can embed '-' in names. (Vincent Ladeuil, #263249)
- Show proper error rather than traceback when an unknown revision id is specified to bzr cat-revision. (Jelmer Vernooij, #175569)

- Trailing text in the dirstate file could cause the C dirstate parser to try to allocate an invalid amount of memory. We now properly check and test for parsing a dirstate with invalid trailing data. (John Arbash Meinel, #186014)
- Unexpected error responses from a smart server no longer cause the client to traceback. (Andrew Bennetts, #263527)
- Use a Windows api function to get a Unicode host name, rather than assuming the host name is ascii. (Mark Hammond, John Arbash Meinel, #256550)
- WorkingTree4 trees will now correctly report missing-and-new paths in the output of iter\_changes. (Robert Collins)

## Documentation

• Updated developer documentation. (Martin Pool)

### **API Changes**

- Exporters now take 4 parameters. (Robert Collins)
- Tree.iter\_changes will now return False for the content change field when a file is missing in the basis tree and not present in the target tree. Previously it returned True unconditionally. (Robert Collins)
- The deprecated Branch.abspath and unimplemented Branch.rename\_one and Branch.move were removed. (Jelmer Vernooij)
- BzrDir.clone\_on\_transport implementations must now accept a stacked\_on parameter. (Aaron Bentley)
- BzrDir.cloning\_metadir implementations must now take a require stacking parameter. (Aaron Bentley)

### **Testing**

- addCleanup now takes \*arguments and \*\*keyword\_arguments which are then passed to the cleanup callable as it is run. In addition, addCleanup no longer requires that the callables passed to it be unique. (Jonathan Lange)
- Fix some tests that fail on Windows because files are deleted while still in use. (Mark Hammond)
- selftest's --starting-with option can now use predefined prefixes so that one can say bzr selftest -s bp.loom instead of bzr selftest -s bzrlib.plugins.loom. (Vincent Ladeuil)
- selftest's --starting-with option now accepts multiple values.
   (Vincent Ladeuil)

#### **Internals**

- A new plugin interface, bzrlib.log.log\_adapters, has been added. This allows dynamic log output filtering by plugins. (Robert Collins)
- bzrlib.btree\_index is now available, providing a b-tree index layer. The design is memory conservative (limited memory cache), faster to seek (approx 100 nodes per page, gives 100way fan out), and stores compressed pages allowing more keys per page. (Robert Collins, John Arbash Meinel)
- bzrlib.diff.DiffTree.show\_diff now skips changes where the kind is unknown in both source and target. (Robert Collins, Aaron Bentley)
- GraphIndexBuilder.add\_node and BTreeBuilder have been streamlined a bit. This should make creating large indexes faster. (In benchmarking, it now takes less time to create a BTree index than it takes to read the GraphIndex one.) (John Arbash Meinel)
- Mail clients for bzr send are now listed in a registry. This allows
  plugins to add new clients by registering them with
  bzrlib.mail\_client.mail\_client\_registry. All of the built-in
  clients now use this mechanism. (Neil Martinsen-Burrell)

### bzr 1.6.1

Released: 2008-09-05

A couple regressions were found in the 1.6 release. There was a performance issue when using bzr+ssh to branch large repositories, and some problems with stacking and rich-root capable repositories.

# bzr 1.6.1rc2

**Released:** 2008-09-03

### **Bug Fixes**

• Copying between rich-root and rich-root-pack (and vice versa) was accidentally using the inter-model fetcher, instead of recognizing that both were 'rich root' formats. (John Arbash Meinel, #264321)

#### bzr 1.6.1rc1

**Released:** 2008-08-29

This release fixes a few regressions found in the 1.6 client. Fetching changes was using an O(N^2) buffering algorithm, so for large projects it would cause memory thrashing. There is also a specific problem with the --1.6-rich-root format, which prevented stacking on top of --rich-root-pack repositories, and could allow users to accidentally fetch experimental data (-subtree) without representing it properly. The --1.6-rich-root format has been deprecated and recommended are to upgrade to users --1.6.1-rich-root immediately. Also we re-introduced a workaround for users who have repositories with incorrect nodes (not possible if you only used official releases). I should also clarify that none of this is data loss level issues, but still sufficient enough to warrant an updated release.

### **Bug Fixes**

- RemoteTransport.readv() was being inefficient about how it buffered the readv data and processed it. It would keep appending to the same string (causing many copies) and then pop bytes out of the start of the string (causing more copies). With this patch "bzr+ssh://local" can improve dramatically, especially for projects with large files. (John Arbash Meinel)
- Revision texts were always meant to be stored as fulltexts.
  There was a bug in a bzr.dev version that would accidentally create deltas when copying from a Pack repo to a Knit repo.
  This has been fixed, but to support those repositories, we know always request full texts for Revision texts. (John Arbash Meinel, #261339)
- The previous --1.6-rich-root format used an incorrect xml serializer, which would accidentally support fetching from a repository that supported subtrees, even though the local one would not. We deprecated that format, and introduced a new one that uses the correct serializer --1.6.1-rich-root. (John Arbash Meinel, #262333)

### bzr 1.6

Released: 2008-08-25

Finally, the long awaited bzr 1.6 has been released. This release includes new features like Stacked Branches, improved weave merge, and an updated server protocol (now on v3) which will allow for better cross version compatibility. With this release we have deprecated Knit format repositories, and recommend that users upgrade them, we will continue to support reading and writing them for the forseeable future, but we will not be tuning them for performance as pack repositories have proven to be better at scaling. This will also be the first release to bundle TortoiseBzr in the standalone Windows installer.

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release
Notes »

previous | next

# bzr 1.6rc5

**Released:** 2008-08-19

### **Bug Fixes**

 Disable automatic detection of stacking based on a containing directory of the target. It interacted badly with push, and needs a bit more work to get the edges polished before it should happen automatically. (John Arbash Meinel, #259275) (This change was reverted when merged to bzr.dev)

# bzr 1.6rc4

**Released:** 2008-08-18

### **Bug Fixes**

• Fix a regression in knit => pack fetching. We had a logic inversion, causing the fetch to insert fulltexts in random order, rather than preserving deltas. (John Arbash Meinel, #256757)

## bzr 1.6rc3

**Released:** 2008-08-14

## Changes

• Disable reading .bzrrules as a per-branch rule preferences file. The feature was not quite ready for a full release. (Robert Collins)

### **Improvements**

• Update the windows installer to bundle TortoiseBzr and qbzr into the standalone installer. This will be the first official windows release that installs Tortoise by default. (Mark Hammond)

### **Bug Fixes**

- Fix a regression in bzr+http support. There was a missing function (\_read\_line) that needed to be carried over from bzr+ssh support. (Andrew Bennetts)
- GraphIndex objects will internally read an entire index if more than 1/20th of their keyspace is requested in a single operation. This largely mitigates a performance regression in bzr log FILE and completely corrects the performance regression in bzr log. The regression was caused by removing an accomodation which had been supporting the index format in use. A newer index format is in development which is substantially faster. (Robert Collins)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release Notes »

previous | next

## bzr 1.6rc2

**Released:** 2008-08-13

This release candidate has a few minor bug fixes, and some regression fixes for Windows.

### **Bug Fixes**

- bzr upgrade on remote branches accessed via bzr:// and bzr+ssh:// now works. (Andrew Bennetts)
- Change the get\_format\_description() strings for RepositoryFormatKnitPack5 et al to be single line messages. (Aaron Bentley)
- Fix for a regression on Win32 where we would try to call os.listdir() on a file and not catch the exception properly. (Windows raises a different exception.) This would manifest in places like bzr rm file or bzr switch. (Mark Hammond, John Arbash Meinel)
- Inventory.copy() was failing to set the revision property for the root entry. (Jelmer Vernooij)
- sftp transport: added missing FileExists case to \_translate\_io\_exception (Christophe Troestler, #123475)
- The help for bzr ignored now suggests bzr 1s --ignored for scripting use. (Robert Collins, #3834)
- The default annotate logic will now always assign the last-modified value of a line to one of the revisions that modified it, rather than a merge revision. This would happen when both sides claimed to have modified the line resulting in the same text. The choice is arbitrary but stable, so merges in different directions will get the same results. (John Arbash Meinel, #232188)

### bzr 1.6rc1

Released: 2008-08-06

This release candidate for bzr 1.6 solidifies the new branch stacking feature. Bazaar now recommends that users upgrade all knit repositories, because later formats are much faster. However, we plan to continue read/write and upgrade support for knit repostories for the forseeable future. Several other bugs and performance issues were fixed.

### Changes

 Knit format repositories are deprecated and bzr will now emit warnings whenever it encounters one. Use bzr upgrade to upgrade knit repositories to pack format. (Andrew Bennetts)

#### **Improvements**

- bzr check can now be told which elements at a location it should check. (Daniel Watkins)
- Commit now supports --exclude (or -x) to exclude some files from the commit. (Robert Collins, #3117)
- Fetching data between repositories that have the same model but no optimised fetcher will not reserialise all the revisions, increasing performance. (Robert Collins, John Arbash Meinel)
- Give a more specific error when target branch is not reachable.
   (James Westby)
- Implemented a custom walkdirs\_utf8 implementation for win32.
   This uses a pyrex extension to get direct access to the FindFirstFilew style apis, rather than using listdir + lstat.
   Shows a very strong improvement in commands like status and diff which have to iterate the working tree. Anywhere from 2x-6x faster depending on the size of the tree (bigger trees, bigger benefit.) (John Arbash Meinel)
- New registry for log properties handles and the method in LongLogFormatter to display the custom properties returned by the registered handlers. (Guillermo Gonzalez, #162469)

#### **Bug Fixes**

- Add more tests that stacking does not create deltas spanning physical repository boundaries. (Martin Pool, #252428)
- Better message about incompatible repositories. (Martin Pool, #206258)
- bzr branch --stacked ensures the destination branch format can support stacking, even if the origin does not. (Martin Pool)
- bzr export no longer exports .bzrrules. (lan Clatworthy)
- bzr serve --directory=/ now correctly allows the whole filesystem to be accessed on Windows, not just the root of the drive that Python is running from. (Adrian Wilkins, #240910)
- Deleting directories by hand before running bzr rm will not cause subsequent errors in bzr st and bzr commit. (Robert Collins, #150438)
- Fix a test case that was failing if encoding wasn't UTF-8. (John Arbash Meinel, #247585)
- Fix "no buffer space available" error when branching with the new smart server protocol to or from Windows. (Andrew Bennetts, #246180)
- Fixed problem in branching from smart server. (#249256, Michael Hudson, Martin Pool)
- Handle a file turning in to a directory in TreeTransform. (James Westby, #248448)

### **API Changes**

- MutableTree.commit has an extra optional keywork parameter exclude that will be unconditionally supplied by the command line UI - plugins that add tree formats may need an update. (Robert Collins)
- The API minimum version for plugin compatibility has been raised to 1.6 there are significant changes throughout the code base. (Robert Collins)
- The generic fetch code now uses three attributes on Repository objects to control fetch. The streams requested are controlled via: \_fetch\_order and \_fetch\_uses\_deltas. Setting these appropriately allows different repository implementations to recieve data in their optimial form. If the \_fetch\_reconcile is set then a reconcile operation is triggered at the end of the fetch. (Robert Collins)
- The put\_on\_disk and get\_tar\_item methods in InventoryEntry were deprecated. (Ian Clatworthy)
- Repository.is\_shared doesn't take a read lock. It didn't need one in the first place (nobody cached the value, and RemoteRepository wasn't taking one either). This saves a round trip when probing Pack repositories, as they read the pack-names file when locked. And during probe, locking the repo isn't very useful. (John Arbash Meinel)

#### **Internals**

- bzrlib.branchbuilder.BranchBuilder is now much more capable of putting together a real history without having to create a full WorkingTree. It is recommended that tests that are not directly testing the WorkingTree use BranchBuilder instead. See BranchBuilder.build\_snapshot or TestCaseWithMemoryTree.make\_branch\_builder. (John Arbash Meinel)
- bzrlib.builtins.internal\_tree\_files broken into two giving a new helper safe\_relpath\_files - used by the new exclude parameter to commit. (Robert Collins)
- Make it easier to introduce new WorkingTree formats. (lan Clatworthy)
- The code for exporting trees was refactored not to use the deprecated InventoryEntry methods. (lan Clatworthy)
- RuleSearchers return () instead of [] now when there are no matches. (Ian Clatworthy)

### bzr 1.6beta3

Released: 2008-07-17

This release adds a new 'stacked branches' feature allowing branches to share storage without being in the same repository or on the same machine. (See the user guide for more details.) It also adds a new hook, improved weaves, aliases for related locations, faster bzr+ssh push, and several bug fixes.

#### **Features**

- New pre\_change\_branch\_tip hook that is called before the branch tip is moved, while the branch is write-locked. See the User Reference for signature details. (Andrew Bennetts)
- Rule-based preferences can now be defined for selected files in selected branches, allowing commands and plugins to provide custom behaviour for files matching defined patterns. See Rulebased preferences (part of Configuring Bazaar) in the User Guide and bzr help rules for more information. (lan Clatworthy)
- Sites may suggest a branch to stack new branches on. (Aaron Bentley)
- Stacked branches are now supported. See bzr help branch and bzr help push. Branches must be in the development1 format to stack, though the stacked-on branch can be of any format. (Robert Collins)

#### **Improvements**

- bzr export --format=tgz --root=NAME to export a gzipped tarball to stdout; also tar and tbz2. (Martin Pool)
- bzr (re)merge --weave will now use a standard Weave algorithm, rather than the annotation-based merge it was using. It does so by building up a Weave of the important texts, without needing to build the full ancestry. (John Arbash Meinel, #238895)
- bzr send documents and better supports emacsclient (proper escaping of mail headers and handling of the MUA Mew). (Christophe Troestler)
- Remembered locations can be specified by aliases, e.g. :parent, :public, :submit. (Aaron Bentley)
- The smart protocol now has improved support for setting branches' revision info directly. This makes operations like push faster. The new request method name is Branch.set\_last\_revision\_ex. (Andrew Bennetts)

#### **Bug Fixes**

- Bazaar is now able to be a client to the web server of IIS 6 and 7. The broken implementations of RFC822 in Python and RFC2046 in IIS combined with boundary-line checking in Bazaar previously made this impossible. (NB, IIS 5 does not suffer from this problem). (Adrian Wilkins, #247585)
- bzr log --long with a ghost in your mainline now handles that ghost properly. (John Arbash Meinel, #243536)
- check handles the split-up .bzr layout correctly, so no longer requires a branch to be present. (Daniel Watkins, #64783)
- Clearer message about how to set the PYTHONPATH if bzrlib can't be loaded. (Martin Pool, #205230)
- Errors about missing libraries are now shown without a traceback, and with a suggestion to install the library. The full traceback is still in .bzr.log and can be shown with -Derror. (Martin Pool, #240161)
- Fetch from a stacked branch copies all required data. (Aaron Bentley, #248506)
- Handle urls such as ftp://user@host.com@www.host.com where the user name contains an @. (Neil Martinsen-Burrell, #228058)
- needs\_read\_lock and needs\_write\_lock now suppress an error during unlock if there was an error in the original function. This helps most when there is a failure with a smart server action, since often the connection closes and we cannot unlock. (Andrew Bennetts, John Arbash Meinel, #125784)
- Obsolete hidden command bzr fetch removed. (Martin Pool, #172870)
- Raise the correct exception when doing -rbefore:0 or -c0. (John Arbash Meinel, #239933)
- You can now compare file revisions in Windows diff programs from Cygwin Bazaar. (Matt McClure, #209281)

- revision\_history now tolerates mainline ghosts for Branch format
  6. (Aaron Bentley, #235055)
- Set locale from environment for third party libs. (Martin von Gagern, #128496)

### Documentation

- Added *Using stacked branches* to the User Guide. (Ian Clatworthy)
- Updated developer documentation. (Martin Pool)

### **Testing**

- -Dmemory will cause /proc/PID/status to be catted before bzr exits, allowing low-key analysis of peak memory use. (Robert Collins)
- TestCaseWithTransport.make\_branch\_and\_tree tries harder to return a tree with a branch attribute of the right format. This was preventing some RemoteBranch tests from actually running with RemoteBranch instances. (Andrew Bennetts)

### **API Changes**

- Removed Repository.text\_store, control\_store, etc. Instead, there are new attributes texts, inventories, revisions, signatures, each of which is a VersionedFiles. See the Repository docstring for more details. (Robert Collins)
- Branch.pull now accepts an \_override\_hook\_target optional parameter. If you have a subclass of Branch that overrides pull then you should add this parameter. (Andrew Bennetts)
- bzrlib.check.check() has been deprecated in favour of the more aptly-named bzrlib.check.check\_branch(). (Daniel Watkins)
- Tree.print\_file and Repository.print\_file are deprecated.
   These methods are bad APIs because they write directly to sys.stdout. bzrlib does not use them internally, and there are no direct tests for them. (Alexander Belchenko)

#### **Internals**

- cat command no longer uses <a href="mailto:tree.print\_file">tree.print\_file()</a> internally. (Alexander Belchenko)
- New class method BzrDir.open\_containing\_tree\_branch\_or\_repository which eases the discovery of the tree, the branch and the repository containing a given location. (Daniel Watkins)
- New versionedfile.KeyMapper interface to abstract out the access to underlying .knit/.kndx etc files in repositories with partitioned storage. (Robert Collins)
- Obsolete developer-use command weave-join has been removed. (Robert Collins)
- RemoteToOtherFetcher and get\_data\_stream\_for\_search removed, to support new VersionedFiles layering. (Robert Collins)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

# bzr 1.6beta2

Released: 2008-06-10

This release contains further progress towards our 1.6 goals of shallow repositories, and contains a fix for some user-affecting bugs in the repository layer. Building working trees during checkout and branch is now faster.

#### **Bug Fixes**

- Avoid KnitCorrupt error extracting inventories from some repositories. (The data is not corrupt; an internal check is detecting a problem reading from the repository.) (Martin Pool, Andrew Bennetts, Robert Collins, #234748)
- bzr status was breaking if you merged the same revision twice.
   (John Arbash Meinel, #235407)
- Fix infinite loop consuming 100% CPU when a connection is lost while reading a response body via the smart protocol v1 or v2. (Andrew Bennetts)
- Inserting a bundle which changes the contents of a file with no trailing end of line, causing a knit snapshot in a 'knits' repository will no longer cause KnitCorrupt. (Robert Collins)
- RemoteBranch.pull needs to return the self.\_real\_branch's pull result. It was instead just returning None, which breaks bzr pull. (John Arbash Meinel, #238149)
- Sanitize branch nick before using it as an attachment filename in bzr send. (Lukáš Lalinský, #210218)
- Squash inv\_entry.symlink\_target to a plain string when generating DirState details. This prevents from getting a UnicodeError when you have symlinks and non-ascii filenames. (John Arbash Meinel, #135320)

### **Improvements**

- Added the 'alias' command to set/unset and display aliases.
   (Tim Penhey)
- added, modified, and unknowns behaviour made consistent (all three now quote paths where required). Added --null option to added and modified (for null-separated unknowns, use 1s -unknown --null) (Adrian Wilkins)
- Faster branching (1.09x) and lightweight checkouts (1.06x) on large trees. (Ian Clatworthy, Aaron Bentley)

# Documentation

• Added Bazaar Zen section to the User Guide. (Ian Clatworthy)

# **Testing**

• Fix the test HTTPServer to be isolated from chdir calls made while it is running, allowing it to be used in blackbox tests. (Robert Collins)

### **API Changes**

- WorkingTree.set\_parent\_(ids/trees) will now filter out revisions which are in the ancestry of other revisions. So if you merge the same tree twice, or merge an ancestor of an existing merge, it will only record the newest. (If you merge a descendent, it will replace its ancestor). (John Arbash Meinel, #235407)
- RepositoryPolicy.\_\_init\_\_ now requires stack\_on and stack\_on\_pwd, through the derived classes do not. (Aaron Bentley)

#### **Internals**

- bzrlib.bzrdir.BzrDir.sprout now accepts stacked to control creating stacked branches. (Robert Collins)
- Knit record serialisation is now stricter on what it will accept, to guard against potential internal bugs, or broken input. (Robert Collins)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

### bzr 1.6beta1

Released: 2008-06-02

Commands that work on the revision history such as push, pull, missing, uncommit and log are now substantially faster. This release adds a translation of some of the user documentation into Spanish. (Contributions of other translations would be very welcome.) Bazaar 1.6beta1 adds a new network protocol which is used by default and which allows for more efficient transfers and future extensions.

### **Notes When Upgrading**

• There is a new version of the network protocol used for bzr://, bzr+ssh:// and bzr+http:// connections. This will allow more efficient requests and responses, and more graceful fallback when a server is too old to recognise a request from a more recent client. Bazaar 1.6 will interoperate with 0.16 and later versions, but servers should be upgraded when possible. Bazaar 1.6 no longer interoperates with 0.15 and earlier via these protocols. Use alternatives like SFTP or upgrade those servers. (Andrew Bennetts, #83935)

# Changes

 Deprecation warnings will not be suppressed when running bzr selftest so that developers can see if their code is using deprecated functions. (John Arbash Meinel)

### **Features**

 Adding -Derror will now display a traceback when a plugin fails to load. (James Westby)

#### **Improvements**

- bzr branch/push/pull -r XXX now have a helper function for finding the revno of the new revision (Graph.find\_distance\_to\_null). This should make something like bzr branch -r -100 in a shared, no-trees repository much snappier. (John Arbash Meinel)
- bzr log --short -r x..y no longer needs to access the full revision history. This makes it noticeably faster when logging the last few revisions. (John Arbash Meinel)
- bzr 1s now accepts -v as an alias for --versioned. (Jerad Cramp, #165086)
- bzr missing uses the new Graph.find\_unique\_ancestors and Graph.find\_differences to determine missing revisions without having to search the whole ancestry. (John Arbash Meinel, #174625)
- bzr uncommit now uses partial history access, rather than always extracting the full revision history for a branch. This makes it resolve the appropriate revisions much faster (in testing it drops uncommit from 1.5s => 0.4s). It also means bzr log --short is one step closer to not using full revision history. (John Arbash Meinel, #172649)

#### **Bugfixes**

- bzr merge --1ca should handle when two revisions have no common ancestor other than NULL\_REVISION. (John Arbash Meinel, #235715)
- bzr status was breaking if you merged the same revision twice.
   (John Arbash Meinel, #235407)
- bzr push with both --overwrite and -r NNN options no longer fails. (Andrew Bennetts, #234229)
- Correctly track the base URL of a smart medium when using bzr+http:// URLs, which was causing spurious "No repository present" errors with branches in shared repositories accessed over bzr+http. (Andrew Bennetts, #230550)
- Define \_remote\_is\_at\_least\_1\_2 on SmartClientMedium so that all implementations have the attribute. Fixes 'PyCurlTransport' object has no attribute '\_remote\_is\_at\_least\_1\_2' attribute errors. (Andrew Bennetts, #220806)
- Failure to delete an obsolete pack file should just give a warning message, not a fatal error. It may for example fail if the file is still in use by another process. (Martin Pool)
- Fix MemoryError during large fetches over HTTP by limiting the amount of data we try to read per recv call. The problem was observed with Windows and a proxy, but might affect other environments as well. (Eric Holmberg, #215426)
- Handle old merge directives correctly in Merger.from\_mergeable. Stricter get\_parent\_map requirements exposed a latent bug here. (Aaron Bentley)
- Issue a warning and ignore passwords declared in authentication.conf when used for an ssh scheme (sftp or bzr+ssh). (Vincent Ladeuil, #203186)
- Make both http implementations raise appropriate exceptions on 403 Forbidden when POSTing smart requests. (Vincent Ladeuil,

#230223)

- Properly *title* header names in http requests instead of capitalizing them. (Vincent Ladeuil, #229076)
- The "Unable to obtain lock" error message now also suggests using bzr break-lock to fix it. (Martin Albisetti, #139202)
- Treat an encoding of "as ascii; this can happen when bzr is run under vim on Mac OS X. (Neil Martinsen-Burrell)
- VersionedFile.make\_mpdiffs() was raising an exception that wasn't in scope. (Daniel Fischer #235687)

#### **Documentation**

- Added directory structure and started translation of docs in spanish. (Martin Albisetti, Lucio Albenga)
- Incorporate feedback from Jelmer Vernooij and Neil Martinsen-Burrell on the plugin and integration chapters of the User Guide. (lan Clatworthy)
- More Bazaar developer documentation about packaging and release process, and about use of Python reprs. (Martin Pool, Martin Albisetti)
- Updated Tortise strategy document. (Mark Hammond)

### **Testing**

- bzrlib.tests.adapt\_tests
   was broken and unused it has been fixed. (Robert Collins)
- Fix the test HTTPServer to be isolated from chdir calls made while it is running, allowing it to be used in blackbox tests. (Robert Collins)
- New helper function for splitting test suites split\_suite\_by\_condition. (Robert Collins)

### Internals

Branch.missing\_revisions has been deprecated. Similar functionality can be obtained using bzrlib.missing.find\_unmerged. The api was fairly broken, and the function was unused, so we are getting rid of it. (John Arbash Meinel)

### **API Changes**

- Branch.abspath is deprecated; use the Tree or Transport instead. (Martin Pool)
- Branch.update\_revisions now takes an optional Graph object.
   This can be used by update\_revisions when it is checking ancestry, and allows callers to prefer request to go to a local branch. (John Arbash Meinel)
- Branch, Repository, Tree and BzrDir should expose a Transport as an attribute if they have one, rather than having it indirectly accessible as .control\_files.\_transport. This doesn't add a requirement to support a Transport in cases where it was not needed before; it just simplifies the way it is reached. (Martin Pool)
- bzr missing --mine-only will return status code 0 if you have no new revisions, but the remote does. Similarly for --theirs-only. The new code only checks one side, so it doesn't know if the other side has changes. This seems more accurate with the request anyway. It also changes the output to print '[This|Other] branch is up to date.' rather than displaying nothing. (John Arbash Meinel)
- LockableFiles.put\_utf8, put\_bytes and controlfilename are now deprecated in favor of using Transport operations. (Martin Pool)
- Many methods on VersionedFile, Repository and in bzrlib.revision deprecated before bzrlib 1.5 have been removed. (Robert Collins)
- RevisionSpec.wants\_revision\_history can be set to False for a given RevisionSpec. This will disable the existing behavior of passing in the full revision history to self.\_match\_on. Useful for specs that don't actually need access to the full history. (John

#### **Arbash Meinel)**

- The constructors of SmartClientMedium and its subclasses now require a base parameter. SmartClientMedium implementations now also need to provide a remote\_path\_from\_transport method. (Andrew Bennetts)
- The default permissions for creating new files and directories should now be obtained from BzrDir.\_get\_file\_mode() and \_get\_dir\_mode(), rather than from LockableFiles. The \_set\_file\_mode and \_set\_dir\_mode variables on LockableFiles which were advertised as a way for plugins to control this are no longer consulted. (Martin Pool)
- VersionedFile.join is deprecated. This method required local instances of both versioned file objects and was thus hostile to being used for streaming from a smart server. The new get\_record\_stream and insert\_record\_stream are meant to efficiently replace this method. (Robert Collins)
- WorkingTree.set\_parent\_(ids/trees) will now filter out revisions which are in the ancestry of other revisions. So if you merge the same tree twice, or merge an ancestor of an existing merge, it will only record the newest. (If you merge a descendent, it will replace its ancestor). (John Arbash Meinel, #235407)
- WorkingTreeFormat2.stub\_initialize\_remote is now private.
   (Martin Pool)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

# bzr 1.5

Released: 2008-05-16

This release of Bazaar includes several updates to the documentation, and fixes to prepare for making rich root support the default format. Many bugs have been squashed, including fixes to log, bzr+ssh inter-operation with older servers.

# Changes

• Suppress deprecation warnings when bzrlib is a 'final' release. This way users of packaged software won't be bothered with DeprecationWarnings, but developers and testers will still see them. (John Arbash Meinel)

### Documentation

• Incorporate feedback from Jelmer Vernooij and Neil Martinsen-Burrell on the plugin and integration chapters of the User Guide. (lan Clatworthy)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

# bzr 1.5rc1

**Released:** 2008-05-09

# Changes

 Broader support of GNU Emacs mail clients. Set mail\_client=emacsclient in your bazaar.conf and send will pop the bundle in a mail buffer according to the value of mail-useragent variable. (Xavier Maillard)

# **Improvements**

- Diff now handles revision specs like "branch:" and "submit:" more efficiently. (Aaron Bentley, #202928)
- More friendly error given when attempt to start the smart server on an address already in use. (Andrea Corbellini, #200575)
- Pull completes much faster when there is nothing to pull. (Aaron Bentley)

#### **Bugfixes**

- Authentication.conf can define sections without password. (Vincent Ladeuil, #199440)
- Avoid muttering every time a child update does not cause a progress bar update. (John Arbash Meinel, #213771)
- Branch.reconcile() is now implemented. This allows bzr reconcile to fix when a Branch has a non-canonical mainline history. bzr check also detects this condition. (John Arbash Meinel, #177855)
- bzr log -r ..x bzr:// was failing, because it was getting a request for revision\_id=None which was not a string. (John Arbash Meinel, #211661)
- bzr commit now works with Microsoft's FTP service. (Andreas Deininger)
- Catch definitions outside sections in authentication.conf. (Vincent Ladeuil, #217650)
- Conversion from non-rich-root to rich-root(-pack) updates inventory sha1s, even when bundles are used. (Aaron Bentley, #181391)
- Conversion from non-rich-root to rich-root(-pack) works correctly even though search keys are not topologically sorted. (Aaron Bentley)
- Conversion from non-rich-root to rich-root(-pack) works even when a parent revision has a different root id. (Aaron Bentley, #177874)
- Disable strace testing until strace is fixed (see bug #103133) and emit a warning when selftest ends to remind us of leaking tests. (Vincent Ladeuil, #226769)
- Fetching all revisions from a repository does not cause pack collisions. (Robert Collins, Aaron Bentley, #212908)
- Fix error about "attempt to add line-delta in non-delta knit".

- (Andrew Bennetts, #217701)
- Pushing a branch in "dirstate" format (Branch5) over bzr+ssh would break if the remote server was < version 1.2. This was due to a bug in the RemoteRepository.get\_parent\_map() fallback code. (John Arbash Meinel, #214894)
- Remove leftover code in bzr\_branch that inappropriately creates a branch-name file in the branch control directory. (Martin Pool)
- Set SO\_REUSEADDR on server sockets of bzr serve to avoid problems rebinding the socket when starting the server a second time. (John Arbash Meinel, Martin Pool, #164288)
- Severe performance degradation in fetching from knit repositories to knits and packs due to parsing the entire revisions.kndx on every graph walk iteration fixed by using the Repository.get\_graph API. There was another regression in knit => knit fetching which re-read the index for every revision each side had in common. (Robert Collins, John Arbash Meinel)
- When logging the changes to a particular file, there was a bug if there were ghosts in the revision ancestry. (John Arbash Meinel, #209948)
- xs4all's ftp server returns a temporary error when trying to list an empty directory, rather than returning an empty list. Adding a workaround so that we don't get spurious failures. (John Arbash Meinel, #215522)

#### Documentation

- Expanded the User Guide to include new chapters on popular plugins and integrating Bazaar into your environment. The *Best practices* chapter was renamed to *Miscellaneous topics* as suggested by community feedback as well. (Ian Clatworthy)
- Document outlining strategies for TortoiseBzr. (Mark Hammond)
- Improved the documentation on hooks. (Ian Clatworthy)
- Update authentication docs regarding ssh agents. (Vincent Ladeuil, #183705)

#### **Testing**

- Add thread\_name\_suffix parameter to SmartTCPServer\_for\_testing, to make it easy to identify which test spawned a thread with an unhandled exception. (Andrew Bennetts)
- New --debugflag/-E option to bzr selftest for setting options for debugging tests, these are complementary to the -D options.
   The -Dselftest\_debug global option has been replaced by the -E=allow\_debug option for selftest. (Andrew Bennetts)
- Parameterised test ids are preserved correctly to aid diagnosis of test failures. (Robert Collins, Andrew Bennetts)
- selftest now accepts –starting-with <id> to load only the tests whose id starts with the one specified. This greatly speeds up running the test suite on a limited set of tests and can be used to run the tests for a single module, a single class or even a single test. (Vincent Ladeuil)
- The test suite modules have been modified to define load\_tests() instead of test\_suite(). That speeds up selective loading (via –load-list) significantly and provides many examples on how to migrate (grep for load\_tests). (Vincent Ladeuil)

#### **Internals**

- Hooks.install\_hook is now deprecated in favour of Hooks.install\_named\_hook which adds a required name parameter, to avoid having to call Hooks.name\_hook. (Daniel Watkins)
- Implement xml8 serializer. (Aaron Bentley)
- New form @deprecated\_method(deprecated\_in(1, 5, 0)) for making deprecation wrappers. (Martin Pool)
- Repository.revision\_parents is now deprecated in favour of Repository.get\_parent\_map([revid])[revid]. (Jelmer Vernooij)
- The Python assert statement is no longer used in Bazaar source, and a test checks this. (Martin Pool)

### **API Changes**

- bzrlib.status.show\_pending\_merges requires the repository to be locked by the caller. Callers should have been doing it anyway, but it will now raise an exception if they do not. (John Arbash Meinel)
- Repository.get\_data\_stream,
   Repository.get\_data\_stream\_for\_search(),
   Repository.get\_deltas\_for\_revsions(),
   Repository.revision\_trees(),
   Repository.item\_keys\_introduced\_by() no longer take read locks. (Aaron Bentley)
- LockableFiles.get\_utf8 and .get are deprecated, as a start towards removing LockableFiles and .control\_files entirely. (Martin Pool)
- Methods deprecated prior to 1.1 have been removed. (Martin Pool)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release

Notes »

previous | next

### bzr 1.4

Released: 2008-04-28

This release of Bazaar includes handy improvements to the speed of log and status, new options for several commands, improved documentation, and better hooks, including initial code for serverside hooks. A number of bugs have been fixed, particularly in interoperability between different formats or different releases of Bazaar over there network. There's been substantial internal work in both the repository and network code to enable new features and faster performance.

# **Bug Fixes**

 Pushing a branch in "dirstate" format (Branch5) over bzr+ssh would break if the remote server was < version 1.2. This was due to a bug in the RemoteRepository.get\_parent\_map() fallback code. (John Arbash Meinel, Andrew Bennetts, #214894)

# bzr 1.4rc2

**Released:** 2008-04-21

#### **Bug Fixes**

- bzr log -r ..x bzr:// was failing, because it was getting a request for revision\_id=None which was not a string. (John Arbash Meinel, #211661)
- Fixed a bug in handling ghost revisions when logging changes in a particular file. (John Arbash Meinel, #209948)
- Fix error about "attempt to add line-delta in non-delta knit". (Andrew Bennetts, #205156)
- Fixed performance degradation in fetching from knit repositories
  to knits and packs due to parsing the entire revisions.kndx on
  every graph walk iteration fixed by using the
  Repository.get\_graph API. There was another regression in knit
  => knit fetching which re-read the index for every revision each
  side had in common. (Robert Collins, John Arbash Meinel)

# bzr 1.4rc1

**Released:** 2008-04-11

# Changes

- bzr main script cannot be imported (Benjamin Peterson)
- On Linux bzr additionally looks for plugins in arch-independent site directory. (Toshio Kuratomi)
- The set\_rh branch hook is now deprecated. Please migrate any plugins using this hook to use an alternative, e.g. post\_change\_branch\_tip. (lan Clatworthy)
- When a plugin cannot be loaded as the file path is not a valid python module name bzr will now strip a bzr\_ prefix from the front of the suggested name, as many plugins (e.g. bzr-svn) want to be installed without this prefix. It is a common mistake to have a folder named "bzr-svn" for that plugin, especially as this is what bzr branch lp:bzr-svn will give you. (James Westby, Andrew Cowie)
- UniqueIntegerBugTracker now appends bug-ids instead of joining them to the base URL. Plugins that register bug trackers may need a trailing / added to the base URL if one is not already there. (James Wesby, Andrew Cowie)

#### **Features**

- Added start\_commit hook for mutable trees. (Jelmer Vernooij, #186422)
- status now accepts --no-pending to show the status without listing pending merges, which speeds up the command a lot on large histories. (James Westby, #202830)
- New post\_change\_branch\_tip hook that is called after the branch tip is moved but while the branch is still write-locked. See the User Reference for signature details. (Ian Clatworthy, James Henstridge)
- Reconfigure can convert a branch to be standalone or to use a shared repository. (Aaron Bentley)

#### **Improvements**

- The smart protocol now has support for setting branches' revision info directly. This should make operations like push slightly faster, and is a step towards server-side hooks. The new request method name is <a href="mailto:Branch.set\_last\_revision\_info">Branch.set\_last\_revision\_info</a>. (Andrew Bennetts)
- bzr commit --fixes now recognises "gnome" as a tag by default. (James Westby, Andrew Cowie)
- bzr switch will attempt to find branches to switch to relative to the current branch. E.g. bzr switch branchname will look for current\_branch/../branchname. (Robert Collins, Jelmer Vernooij, Wouter van Heyst)
- Diff is now more specific about execute-bit changes it describes (Chad Miller)
- Fetching data over HTTP is a bit faster when urllib is used. This is done by forcing it to recv 64k at a time when reading lines in HTTP headers, rather than just 1 byte at a time. (Andrew Bennetts)
- Log –short and –line are much faster when -r is not specified.
   (Aaron Bentley)
- Merge is faster. We no longer check a file's existence unnecessarily when merging the execute bit. (Aaron Bentley)
- bzr status on an explicit list of files no longer shows pending merges, making it much faster on large trees. (John Arbash Meinel)
- The launchpad directory service now warns the user if they have not set their launchpad login and are trying to resolve a URL using it, just in case they want to do a write operation with it. (James Westby)
- The smart protocol client is slightly faster, because it now only queries the server for the protocol version once per connection.

Also, the HTTP transport will now automatically probe for and use a smart server if one is present. You can use the new nosmart+ transport decorator to get the old behaviour. (Andrew Bennetts)

- The version command takes a --short option to print just the version number, for easier use in scripts. (Martin Pool)
- Various operations with revision specs and commands that calculate revnos and revision ids are faster. (John A. Meinel, Aaron Bentley)

#### **Bugfixes**

- Add root\_client\_path parameter to SmartWSGIApp and SmartServerRequest. This makes it possible to publish filesystem locations that don't exactly match URL paths. SmartServerRequest subclasses should use the new translate\_client\_path and transport\_from\_client\_path methods when dealing with paths received from a client to take this into account. (Andrew Bennetts, #124089)
- bzr mv a b can be now used also to rename previously renamed directories, not only files. (Lukáš Lalinský, #107967)
- bzr uncommit --local can now remove revisions from the local branch to be symmetric with bzr commit --local. (John Arbash Meinel, #93412)
- Don't ask for a password if there is no real terminal. (Alexander Belchenko, #69851)
- Fix a bug causing a ValueError crash in parse\_line\_delta\_iter
  when fetching revisions from a knit to pack repository or vice
  versa using bzr:// (including over http or ssh). (#208418, Andrew
  Bennetts, Martin Pool, Robert Collins)
- Fixed \_get\_line in bzrlib.smart.medium, which was buggy. Also fixed \_get\_bytes in the same module to use the push back buffer. These bugs had no known impact in normal use, but were problematic for developers working on the code, and were likely to cause real bugs sooner or later. (Andrew Bennetts)
- Implement handling of basename parameter for DefaultMail. (James Westby)
- Incompatibility with Paramiko versions newer than 1.7.2 was fixed. (Andrew Bennetts, #213425)
- Launchpad locations (lp: URLs) can be pulled. (Aaron Bentley, #181945)
- Merges that add files to deleted root directories complete. They

do create conflicts. (Aaron Bentley, #210092)

• vsftp's return 550 RNFR command failed. supported. (Marcus Trautwig, #129786)

#### Documentation

- Improved documentation on send/merge relationship. (Peter Schuller)
- Minor fixes to the User Guide. (Matthew Fuller)
- Reduced the evangelism in the User Guide. (Ian Clatworthy)
- Added Integrating with Bazaar document for developers (Martin Albisetti)

#### **API Breaks**

- Attempting to pull data from a ghost aware repository (e.g. knits) into a non-ghost aware repository such as weaves will now fail if there are ghosts. (Robert Collins)
- KnitVersionedFile no longer accepts an access\_mode parameter, and now requires the index and access\_method parameters to be supplied. A compatible shim has been kept in the new function knit.make\_file\_knit. (Robert Collins)
- Log formatters must now provide log\_revision instead of show and show\_merge\_revno methods. The latter had been deprecated since the 0.17 release. (James Westby)
- LoopbackSFTP is now called SocketAsChannelAdapter. (Andrew Bennetts)
- osutils.backup\_file is removed. (Alexander Belchenko)
- Repository.get\_revision\_graph is deprecated, with no replacement method. The method was size(history) and not desirable. (Robert Collins)
- revision.revision\_graph is deprecated, with no replacement function. The function was size(history) and not desirable. (Robert Collins)
- Transport.get\_shared\_medium is deprecated. Use
   Transport.get\_smart\_medium instead. (Andrew Bennetts)
- VersionedFile factories now accept a get\_scope parameter rather than using a call to transaction\_finished, allowing the removal of the fixed list of versioned files per repository. (Robert Collins)
- VersionedFile.annotate\_iter is deprecated. While in principle this allowed lower memory use, all users of annotations wanted full file annotations, and there is no storage format suitable for incremental line-by-line annotation. (Robert Collins)

- VersionedFile.clone\_text is deprecated. This performance optimisation is no longer used - reading the content of a file that is undergoing a file level merge to identical state on two branches is rare enough, and not expensive enough to special case. (Robert Collins)
- VersionedFile.clear\_cache and enable\_cache are deprecated. These methods added significant complexity to implementation, but were VersionedFile only used for optimising fetches from knits - which can be done from outside the knit layer, or via a caching decorator. As knits are not the default format, the complexity is no longer worth paying. (Robert Collins)
- VersionedFile.create\_empty is removed. This method presupposed a sensible mapping to a transport for individual files, but pack backed versioned files have no such mapping. (Robert Collins)
- VersionedFile.get\_graph is deprecated, with no replacement method. The method was size(history) and not desirable. (Robert Collins)
- VersionedFile.get\_graph\_with\_ghosts is deprecated, with no replacement method. The method was size(history) and not desirable. (Robert Collins)
- VersionedFile.get\_parents is deprecated, please use
   VersionedFile.get\_parent\_map. (Robert Collins)
- VersionedFile.get\_sha1 is deprecated, please use
   VersionedFile.get\_sha1s. (Robert Collins)
- VersionedFile.has\_ghost is now deprecated, as it is both expensive and unused outside of a single test. (Robert Collins)
- VersionedFile.iter\_parents is now deprecated in favour of get\_parent\_map which can be used to instantiate a Graph on a VersionedFile. (Robert Collins)
- VersionedFileStore no longer uses the transaction parameter given to most methods; amongst other things this means that

the get\_weave\_or\_empty method no longer guarantees errors on a missing weave in a readonly transaction, and no longer caches versioned file instances which reduces memory pressure (but requires more careful management by callers to preserve performance). (Robert Collins)

### **Testing**

- New -Dselftest\_debug flag disables clearing of the debug flags during tests. This is useful if you want to use e.g. -Dhpss to help debug a failing test. Be aware that using this feature is likely to cause spurious test failures if used with the full suite. (Andrew Bennetts)
- selftest –load-list now uses a new more agressive test loader that will avoid loading unneeded modules and building their tests. Plugins can use this new loader by defining a load\_tests function instead of a test\_suite function. (a forthcoming patch will provide many examples on how to implement this). (Vincent Ladeuil)
- selftest –load-list now does some sanity checks regarding duplicate test IDs and tests present in the list but not found in the actual test suite. (Vincent Ladeuil)
- Slightly more concise format for the selftest progress bar, so there's more space to show the test name. (Martin Pool)

```
[2500/10884, 1fail, 3miss in 1m29s] test_revisionnamespaces
```

 The test suite takes much less memory to run, and is a bit faster. This is done by clearing most attributes of TestCases after running them, if they succeeded. (Andrew Bennetts)

#### Internals

- Added \_build\_client\_protocol to \_SmartClient. (Andrew Bennetts)
- Added basic infrastructure for automatic plugin suggestion. (Martin Albisetti)
- If a LockableFiles object is not explicitly unlocked (for example because of a missing try/finally block, it will give a warning but not automatically unlock itself. (Previously they did.) This sometimes caused knock-on errors if for example the network connection had already failed, and should not be relied upon by code. (Martin Pool, #109520)
- make dist target to build a release tarball, and also check-disttarball and dist-upload-escudero. (Martin Pool)
- The read\_response\_tuple method of SmartClientRequestProtocol\* classes will now raise UnknownSmartMethod when appropriate, so that callers don't need to try distinguish unknown request errors from other errors. (Andrew Bennetts)
- set\_make\_working\_trees is now implemented provided on all
  repository implementations (Aaron Bentley)
- VersionedFile now has a new method get\_parent\_map which, like Graph.get\_parent\_map returns a dict of key:parents. (Robert Collins)

# bzr 1.3.1

**Released:** 2008-04-09

No changes from 1.3.1rc1.

# bzr 1.3.1rc1

Released: 2008-04-04

# **Bug Fixes**

Fix a bug causing a ValueError crash in parse\_line\_delta\_iter when fetching revisions from a knit to pack repository or vice versa using bzr:// (including over http or ssh). (#208418, Andrew Bennetts, Martin Pool, Robert Collins)

# bzr 1.3

Released: 2008-03-20

Bazaar has become part of the GNU project <a href="http://www.gnu.org">http://www.gnu.org</a>

Many operations that act on history, including log and annotate are now substantially faster. Several bugs have been fixed and several new options and features have been added.

# **Testing**

 Avoid spurious failure of TestVersion.test\_version matching directory names. (#202778, Martin Pool)

# bzr 1.3rc1

**Released:** 2008-03-16

## **Notes When Upgrading**

• The backup directory created by upgrade is now called backup.bzr, not .bzr.backup. (Martin Albisetti)

## Changes

- A new repository format 'development' has been added. This format will represent the latest 'in-progress' format that the bzr developers are interested in getting early-adopter testing and feedback on. doc/developers/development-repo.txt has detailed information. (Robert Collins)
- BZR\_LOG environment variable controls location of .bzr.log trace file. User can suppress writing messages to .bzr.log by using '/dev/null' filename (on Linux) or 'NUL' (on Windows). If BZR\_LOG variable is not defined but BZR\_HOME is defined then default location for .bzr.log trace file is \$BZR\_HOME/.bzr.log. (Alexander Belchenko, #106117)
- launchpad builtin plugin now shipped as separate part in standalone bzr.exe, installed to C:\Program Files\Bazaar\plugins directory, and standalone installer allows user to skip installation of this plugin. (Alexander Belchenko)
- Restore auto-detection of plink.exe on Windows. (Dmitry Vasiliev)
- Version number is now shown as "1.2" or "1.2pr2", without zeroed or missing final fields. (Martin Pool)

### **Features**

- branch and checkout can hard-link working tree files, which is faster and saves space. (Aaron Bentley)
- bzr send will now also look at the child\_submit\_to setting in the submit branch to determine the email address to send to. (Jelmer Vernooij)

#### **Improvements**

- BzrBranch.\_lefthand\_history is faster on pack repos. (Aaron Bentley)
- Branch6.generate\_revision\_history is faster. (Aaron Bentley)
- Directory services can now be registered, allowing special URLs to be dereferenced into real URLs. This is a generalization and cleanup of the lp: transport lookup. (Aaron Bentley)
- Merge directives that are automatically attached to emails have nicer filenames, based on branch-nick + revno. (Aaron Bentley)
- push has a --revision option, to specify what revision to push up to. (Daniel Watkins)
- Significantly reducing execution time and network traffic for trivial case of running bzr missing command for two identical branches. (Alexander Belchenko)
- Speed up operations that look at the revision graph (such as 'bzr log'). KnitPackRepositor.get\_revision\_graph uses Graph.iter\_ancestry to extract the revision history. This allows filtering ghosts while stepping instead of needing to peek ahead. (John Arbash Meinel)
- The hooks command lists installed hooks, to assist in debugging. (Daniel Watkins)
- Updates to how annotate work. Should see a measurable improvement in performance and memory consumption for file with a lot of merges. Also, correctly handle when a line is introduced by both parents (it should be attributed to the first merge which notices this, and not to all subsequent merges.) (John Arbash Meinel)

### **Bugfixes**

- Autopacking no longer holds the full set of inventory lines in memory while copying. For large repositories, this can amount to hundreds of MB of ram consumption. (Ian Clatworthy, John Arbash Meinel)
- Cherrypicking when using --format=merge3 now explictly excludes BASE lines. (John Arbash Meinel, #151731)
- Disable plink's interactive prompt for password. (#107593, Dmitry Vasiliev)
- Encode command line arguments from unicode to user\_encoding before invoking external mail client in bzr send command. (#139318, Alexander Belchenko)
- Fixed problem connecting to bzr+https:// servers. (#198793, John Ferlito)
- Improved error reporting in the Launchpad plugin. (Daniel Watkins, #196618)
- Include quick-start-summary.svg file to python-based installer(s) for Windows. (#192924, Alexander Belchenko)
- Ica merge now respects specified files. (Aaron Bentley)
- Make version-info –custom imply –all. (#195560, James Westby)
- merge --preview now works for merges that add or modify symlinks (James Henstridge)
- Redirecting the output from bzr merge (when the remembered location is used) now works. (John Arbash Meinel)
- setup.py script explicitly checks for Python version. (Jari Aalto, Alexander Belchenko, #200569)
- UnknownFormatErrors no longer refer to branches regardless of kind of unknown format. (Daniel Watkins, #173980)
- Upgrade bundled ConfigObj to version 4.5.2, which properly quotes # signs, among other small improvements. (Matt

Nordhoff, #86838)

• Use correct indices when emitting LCA conflicts. This fixes IndexError errors. (Aaron Bentley, #196780)

## Documentation

• Explained how to use version-info --custom in the User Guide. (Neil Martinsen-Burrell)

## **API Breaks**

 Support for loading plugins from zip files and bzrlib.plugin.load\_from\_zip() function are deprecated. (Alexander Belchenko)

## **Testing**

- Added missing blackbox tests for modified (Adrian Wilkins)
- The branch interface tests were invalid for branches using richroot repositories because the empty string is not a valid file-id. (Robert Collins)

#### **Internals**

- Graph.iter\_ancestry returns the ancestry of revision ids. Similar to Repository.get\_revision\_graph() except it includes ghosts and you can stop part-way through. (John Arbash Meinel)
- New module tools/package\_mf.py provide custom module finder for python packages (improves standard python library's modulefinder.py) used by setup.py script while building standalone bzr.exe. (Alexander Belchenko)
- New remote method RemoteBzrDir.find\_repositoryv2 adding support for detecting external lookup support on remote repositories. This method is now attempted first when lookup up repositories, leading to an extra round trip on older bzr smart servers. (Robert Collins)
- Repository formats have a new supported-feature attribute supports\_external\_lookups used to indicate repositories which support falling back to other repositories when they have partial data. (Robert Collins)
- Repository.get\_revision\_graph\_with\_ghosts and bzrlib.revision.
   (common\_ancestor, MultipleRevisionSources, common\_graph) have been deprecated. (John Arbash Meinel)
- Tree.iter\_changes is now a public API, replacing the work-in-progress Tree.\_iter\_changes. The api is now considered stable and ready for external users. (Aaron Bentley)
- The bzrdir format registry now accepts an alias keyword to register\_metadir, used to indicate that a format name is an alias for some other format and thus should not be reported when describing the format. (Robert Collins)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

# bzr 1.2

**Released:** 2008-02-15

## **Bug Fixes**

• Fix failing test in Launchpad plugin. (Martin Pool)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

# bzr 1.2rc1

**Released:** 2008-02-13

## Notes When Upgrading

• Fetching via the smart protocol may need to reconnect once during a fetch if the remote server is running Bazaar 1.1 or earlier, because the client attempts to use more efficient requests that confuse older servers. You may be required to reenter a password or passphrase when this happens. This won't happen if the server is upgraded to Bazaar 1.2. (Andrew Bennetts)

## Changes

- Fetching via bzr+ssh will no longer fill ghosts by default (this is consistent with pack-0.92 fetching over SFTP). (Robert Collins)
- Formatting of bzr plugins output is changed to be more humanfriendly. Full path of plugins locations will be shown only with -verbose command-line option. (Alexander Belchenko)
- merge now prefers to use the submit branch, but will fall back to parent branch. For many users, this has no effect. But some users who pull and merge on the same branch will notice a change. This change makes it easier to work on a branch on two different machines, pulling between the machines, while merging from the upstream. merge --remember can now be used to set the submit branch. (Aaron Bentley)

#### **Features**

- merge --preview produces a diff of the changes merge would make, but does not actually perform the merge. (Aaron Bentley)
- New smart method Repository.get\_parent\_map for getting revision parent data. This returns additional parent information topologically adjacent to the requested data to reduce round trip latency impacts. (Robert Collins)
- New smart method, Repository.stream\_revisions\_chunked, for fetching revision data that streams revision data via a chunked encoding. This avoids buffering large amounts of revision data on the server and on the client, and sends less data to the server to request the revisions. (Andrew Bennetts, Robert Collins, #178353)
- The launchpad plugin now handles Ip urls of the form <a href="lip://staging/">1p://demo/</a>, <a href="lip://dev/">1p://dev/</a> to use the appropriate launchpad instance to do the resolution of the branch identities. This is primarily of use to Launchpad developers, but can also be used by other users who want to try out Launchpad as a branch location without messing up their public Launchpad account. Branches that are pushed to the staging environment have an expected lifetime of one day. (Tim Penhey)

### **Improvements**

- Creating a new branch no longer tries to read the entire revision-history unnecessarily over smart server operations. (Robert Collins)
- Fetching between different repository formats with compatible models now takes advantage of the smart method to stream revisions. (Andrew Bennetts)
- The --coverage option is now global, rather specific to bzr selftest. (Andrew Bennetts)
- The register-branch command will now use the public url of the branch containing the current directory, if one has been set and no explicit branch is provided. (Robert Collins)
- Tweak the reannotate code path to optimize the 2-parent case. Speeds up bzr annotate with a pack repository by approx 3:2. (John Arbash Meinel)

### **Bugfixes**

- Calculate remote path relative to the shared medium in \_SmartClient. This is related to the problem in bug #124089. (Andrew Bennetts)
- Cleanly handle connection errors in smart protocol version two, the same way as they are handled by version one. (Andrew Bennetts)
- Clearer error when version-info --custom is used without --template (Lukáš Lalinský)
- Don't raise UnavailableFeature during test setup when medusa is not available or tearDown is never called leading to nasty side effects. (#137823, Vincent Ladeuil)
- If a plugin's test suite cannot be loaded, for example because of a syntax error in the tests, then selftest fails, rather than just printing a warning. (Martin Pool, #189771)
- List possible values for BZR\_SSH environment variable in envvariables help topic. (Alexander Belchenko, #181842)
- New methods push\_log\_file and pop\_log\_file to intercept messages: popping the log redirection now precisely restores the previous state, which makes it easier to use bzr log output from other programs. TestCaseInTempDir no longer depends on a log redirection being established by the test framework, which lets bzr tests cleanly run from a normal unittest runner. (#124153, #124849, Martin Pool, Jonathan Lange)
- pull --quiet is now more quiet, in particular a message is no longer printed when the remembered pull location is used. (James Westby, #185907)
- reconfigure can safely be interrupted while fetching. (Aaron Bentley, #179316)
- reconfigure preserves tags when converting to and from lightweight checkouts. (Aaron Bentley, #182040)

- Stop polluting /tmp when running selftest. (Vincent Ladeuil, #123363)
- Switch from NFKC => NFC for normalization checks. NFC allows a few more characters which should be considered valid. (John Arbash Meinel, #185458)
- The launchpad plugin now uses the edge xmlrpc server to avoid interacting badly with a bug on the launchpad side. (Robert Collins)
- Unknown hostnames when connecting to a bzr:// URL no longer cause tracebacks. (Andrew Bennetts, #182849)

### **API Breaks**

- Classes implementing Merge types like Merge3Merger must now accept (and honour) a do\_merge flag in their constructor. (Aaron Bentley)
- Repository.add\_inventory and add\_revision now require the caller to previously take a write lock (and start a write group.) (Martin Pool)

## **Testing**

 selftest now accepts –load-list <file> to load a test id list. This speeds up running the test suite on a limited set of tests. (Vincent Ladeuil)

#### Internals

- Add a new method get\_result to graph search objects. The resulting SearchResult can be used to recreate the search later, which will be useful in reducing network traffic. (Robert Collins)
- Use convenience function to check whether two repository handles are referring to the same repository in Repository.get\_graph. (Jelmer Vernooij, #187162)
- Fetching now passes the find\_ghosts flag through to the InterRepository.missing\_revision\_ids call consistently for all repository types. This will enable faster missing revision discovery with bzr+ssh. (Robert Collins)
- Fix error handling in Repository.insert\_data\_stream. (Lukas Lalinsky)
- InterRepository.missing\_revision\_ids is now deprecated in favour of InterRepository.search\_missing\_revision\_ids which returns a bzrlib.graph.SearchResult suitable for making requests from the smart server. (Robert Collins)
- New error NoPublicBranch for commands that need a public branch to operate. (Robert Collins)
- New method iter\_inventories on Repository for access to many inventories. This is primarily used by the revision\_trees method, as direct access to inventories is discouraged. (Robert Collins)
- New method next\_with\_ghosts on the Graph breadth-firstsearch objects which will split out ghosts and present parents into two separate sets, useful for code which needs to be aware of ghosts (e.g. fetching data cares about ghosts during revision selection). (Robert Collins)
- Record a timestamp against each mutter to the trace file, relative to the first import of bzrlib. (Andrew Bennetts)

- Repository.get\_data\_stream is now deprecated in favour of Repository.get\_data\_stream\_for\_search which allows less network traffic when requesting data streams over a smart server. (Robert Collins)
- RemoteBzrDir.\_get\_tree\_branch no longer triggers \_ensure\_real, removing one round trip on many network operations. (Robert Collins)
- RemoteTransport's recommended\_page\_size method now returns 64k, like SFTPTransport and HttpTransportBase. (Andrew Bennetts)
- Repository has a new method has\_revisions which signals the
  presence of many revisions by returning a set of the revisions
  listed which are present. This can be done by index queries
  without reading data for parent revision names etc. (Robert
  Collins)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

## bzr 1.1

**Released:** 2008-01-15

(no changes from 1.1rc1)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

## bzr 1.1rc1

**Released:** 2008-01-05

### Changes

- Dotted revision numbers have been revised. Instead of growing longer with nested branches the branch number just increases. (eg instead of 1.1.1.1.1 we now report 1.2.1.) This helps scale long lived branches which have many feature branches merged between them. (John Arbash Meinel)
- The syntax bzr diff branch1 branch2 is no longer supported. Use bzr diff branch1 --new branch2 instead. This change has been made to remove the ambiguity where branch2 is in fact a specific file to diff within branch1.

#### **Features**

- New option to use custom template-based formats in bzr version-info. (Lukáš Lalinský)
- diff '-using' allows an external diff tool to be used for files. (Aaron Bentley)
- New "Ica" merge-type for fast everyday merging that also supports criss-cross merges. (Aaron Bentley)

#### **Improvements**

- annotate now doesn't require a working tree. (Lukáš Lalinský, #90049)
- branch and checkout can now use files from a working tree to to speed up the process. For checkout, this requires the new – files-from flag. (Aaron Bentley)
- bzr diff now sorts files in alphabetical order. (Aaron Bentley)
- bzr diff now works on branches without working trees. Tree-less branches can also be compared to each other and to working trees using the new diff options --old and --new. Diffing between branches, with or without trees, now supports specific file filtering as well. (Ian Clatworthy, #6700)
- bzr pack now orders revision texts in topological order, with newest at the start of the file, promoting linear reads for bzr log and the like. This partially fixes #154129. (Robert Collins)
- Merge directives now fetch prerequisites from the target branch if needed. (Aaron Bentley)
- pycurl now handles digest authentication. (Vincent Ladeuil)
- reconfigure can now convert from repositories. (Aaron Bentley)
- -1 is now a short form for --limit in log. (Matt Nordhoff)
- merge now warns when merge directives cause cherrypicks.
   (Aaron Bentley)
- split now supported, to enable splitting large trees into smaller pieces. (Aaron Bentley)

### **Bugfixes**

- Avoid AttributeError when unlocking a pack repository when an error occurs. (Martin Pool, #180208)
- Better handle short reads when processing multiple range requests. (Vincent Ladeuil, #179368)
- build\_tree acceleration uses the correct path when a file has been moved. (Aaron Bentley)
- commit now succeeds when a checkout and its master branch share a repository. (Aaron Bentley, #177592)
- Fixed error reporting of unsupported timezone format in log -- timezone. (Lukáš Lalinský, #178722)
- Fixed Unicode encoding error in ignored when the output is redirected to a pipe. (Lukáš Lalinský)
- Fix traceback when sending large response bodies over the smart protocol on Windows. (Andrew Bennetts, #115781)
- Fix urlutils.relative\_url for the case of two file:/// URLs pointed to different logical drives on Windows. (Alexander Belchenko, #90847)
- HTTP test servers are now compatible with the http protocol version 1.1. (Vincent Ladeuil, #175524)
- \_KnitParentsProvider.get\_parent\_map now handles requests for ghosts correctly, instead of erroring or attributing incorrect parents to ghosts. (Aaron Bentley)
- merge --weave --uncommitted now works. (Aaron Bentley)
- pycurl authentication handling was broken and incomplete. Fix handling of user:pass embedded in the urls. (Vincent Ladeuil, #177643)
- Files inside non-directories are now handled like other conflict types. (Aaron Bentley, #177390)
- reconfigure is able to convert trees into lightweight checkouts.
   (Aaron Bentley)

- Reduce lockdir timeout to 0 when running bzr serve. (Andrew Bennetts, #148087)
- Test that the old <a href="version\_info\_format">version\_info\_format</a> functions still work, even though they are deprecated. (John Arbash Meinel, ShenMaq, #177872)
- Transform failures no longer cause ImmortalLimbo errors (Aaron Bentley, #137681)
- uncommit works even when the commit messages of revisions to be removed use characters not supported in the terminal encoding. (Aaron Bentley)
- When dumb http servers return whole files instead of the requested ranges, read the remaining bytes by chunks to avoid overflowing network buffers. (Vincent Ladeuil, #175886)

#### **Documentation**

- Minor tweaks made to the bug tracker integration documentation. (lan Clatworthy)
- Reference material has now be moved out of the User Guide and added to the User Reference. The User Reference has gained 4 sections as a result: Authenication Settings, Configuration Settings, Conflicts and Hooks. All help topics are now dumped into text format in the doc/en/user-reference directory for those who like browsing that information in their editor. (Ian Clatworthy)
- Using Bazaar with Launchpad tutorial added. (Ian Clatworthy)

#### **Internals**

- find\_\* methods available for BzrDirs, Branches and WorkingTrees. (Aaron Bentley)
- Help topics can now be loaded from files. (Ian Clatworthy, Alexander Belchenko)
- get\_parent\_map now always provides tuples as its output. (Aaron Bentley)
- Parent Providers should now implement get\_parent\_map returning a dictionary instead of get\_parents returning a list. Graph.get\_parents is now deprecated. (John Arbash Meinel, Robert Collins)
- Patience Diff now supports arbitrary python objects, as long as they support hash(). (John Arbash Meinel)
- Reduce selftest overhead to establish test names by memoization. (Vincent Ladeuil)

## **API Breaks**

### **Testing**

- Modules can now customise their tests by defining a load\_tests attribute.
   pydoc
   bzrlib.tests.TestUtil.TestLoader.loadTestsFromModule for the documentation on this attribute. (Robert Collins)
- New helper function bzrlib.tests.condition\_id\_re which helps filter tests based on a regular expression search on the tests id. (Robert Collins)
- New helper function bzrlib.tests.condition\_isinstance which helps filter tests based on class. (Robert Collins)
- New helper function bzrlib.tests.exclude\_suite\_by\_condition which generalises the exclude\_suite\_by\_re function. (Robert Collins)
- New helper function bzrlib.tests.filter\_suite\_by\_condition which generalises the filter\_suite\_by\_re function. (Robert Collins)
- New helper method <a href="bzrlib.tests.exclude\_tests\_by\_re">bzrlib.tests.exclude\_tests\_by\_re</a> which gives a new TestSuite that does not contain tests from the input that matched a regular expression. (Robert Collins)
- New helper method bzrlib.tests.randomize\_suite which returns a randomized copy of the input suite. (Robert Collins)
- New helper method bzrlib.tests.split\_suite\_by\_re which splits a test suite into two according to a regular expression. (Robert Collins)
- Parametrize all http tests for the transport implementations, the http protocol versions (1.0 and 1.1) and the authentication schemes. (Vincent Ladeuil)
- The exclude\_pattern and random\_order parameters to the function bzrlib.tests.filter\_suite\_by\_re have been deprecated. (Robert Collins)

• The method <a href="mailto:bzrlib.tests.sort\_suite\_by\_re">bzrlib.tests.sort\_suite\_by\_re</a> has been deprecated. It is replaced by the new helper methods added in this release. (Robert Collins)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

# bzr 1.0

**Released:** 2007-12-14

#### **Documentation**

- More improvements and fixes to the User Guide. (lan Clatworthy)
- Add information on cherrypicking/rebasing to the User Guide. (lan Clatworthy)
- Improve bug tracker integration documentation. (Ian Clatworthy)
- Minor edits to Bazaar in five minutes from David Roberts and to the rebasing section of the User Guide from Aaron Bentley. (lan Clatworthy)

previous | next

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release

Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

# bzr 1.0rc3

**Released:** 2007-12-11

## Changes

• If a traceback occurs, users are now asked to report the bug through Launchpad (https://bugs.launchpad.net/bzr/), rather than by mail to the mailing list. (Martin Pool)

## **Bugfixes**

- Fix Makefile rules for doc generation. (Ian Clatworthy, #175207)
- Give more feedback during long http downloads by making readv deliver data as it arrives for urllib, and issue more requests for pycurl. High latency networks are better handled by urllib, the pycurl implementation give more feedback but also incur more latency. (Vincent Ladeuil, #173010)
- Implement \_make\_parents\_provider on RemoteRepository, allowing generating bundles against branches on a smart server. (Andrew Bennetts, #147836)

## Documentation

- Improved user guide. (lan Clatworthy)
- The single-page quick reference guide is now available as a PDF. (Ian Clatworthy)

## **Internals**

• readv urllib http implementation is now a real iterator above the underlying socket and deliver data as soon as it arrives. 'get' still wraps its output in a StringIO. (Vincent Ladeuil)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

# bzr 1.0rc2

**Released:** 2007-12-07

#### **Improvements**

- Added a –coverage option to selftest. (Andrew Bennetts)
- Annotate merge (merge-type=weave) now supports cherrypicking. (Aaron Bentley)
- bzr commit now doesn't print the revision number twice. (Matt Nordhoff, #172612)
- New configuration option bugtracker\_<tracker\_abbrevation>\_url to define locations of bug trackers that are not directly supported by bzr or a plugin. The URL will be treated as a template and {id} placeholders will be replaced by specific bug IDs. (Lukáš Lalinský)
- Support logging single merge revisions with short and line log formatters. (Kent Gibson)
- User Guide enhanced with suggested readability improvements from Matt Revell and corrections from John Arbash Meinel. (Ian Clatworthy)
- Quick Start Guide renamed to Quick Start Card, moved down in the catalog, provided in pdf and png format and updated to refer to send instead of bundle. (lan Clatworthy, #165080)
- switch can now be used on heavyweight checkouts as well as lightweight ones. After switching a heavyweight checkout, the local branch is a mirror/cache of the new bound branch and uncommitted changes in the working tree are merged. As a safety check, if there are local commits in a checkout which have not been committed to the previously bound branch, then switch fails unless the --force option is given. This option is now also required if the branch a lightweight checkout is pointing to has been moved. (Ian Clatworthy)

## Internals

- New -Dhttp debug option reports http connections, requests and responses. (Vincent Ladeuil)
- New -Dmerge debug option, which emits merge plans for merge-type=weave.

## **Bugfixes**

- Better error message when running bzr cat on a non-existant branch. (Lukáš Lalinský, #133782)
- Catch OSError 17 (file exists) in final phase of tree transform and show filename to user. (Alexander Belchenko, #111758)
- Catch ShortReadvErrors while using pycurl. Also make readv more robust by allowing multiple GET requests to be issued if too many ranges are required. (Vincent Ladeuil, #172701)
- Check for missing basis texts when fetching from packs to packs. (John Arbash Meinel, #165290)
- Fall back to showing e-mail in log --short/--line if the committer/author has only e-mail. (Lukáš Lalinský, #157026)

## **API Breaks**

• Deprecate not passing a location argument to commit reporters' started methods. (Matt Nordhoff)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

# bzr 1.0rc1

**Released:** 2007-11-30

## **Notes When Upgrading**

The default repository format is now pack-0.92. This default is used when creating new repositories with init and init-repo, and when branching over bzr+ssh or bzr+hpss. (See https://bugs.launchpad.net/bugs/164626)

This format can be read and written by Bazaar 0.92 and later, and data can be transferred to and from older formats.

To upgrade, please reconcile your repository (bzr reconcile), and then upgrade (bzr upgrade).

pack-0.92 offers substantially better scaling and performance than the previous knits format. Some operations are slower where the code already had bad scaling characteristics under knits, the pack format makes such operations more visible as part of being more scalable overall. We will correct such operations over the coming releases and encourage the filing of bugs on any operation which you observe to be slower in a packs repository. One particular case that we do not intend to fix is pulling data from a pack repository into a knit repository over a high latency link; downgrading such data requires reinsertion of the file texts, and this is a classic space/time tradeoff. The current implementation is conservative on memory usage because we need to support converting data from any tree without problems. (Robert Collins, Martin Pool, #164476)

## Changes

- Disable detection of plink.exe as possible ssh vendor. Plink vendor still available if user selects it explicitly with BZR\_SSH environment variable. (Alexander Belchenko, workaround for bug #107593)
- The pack format is now accessible as "pack-0.92", or "pack-0.92-subtree" to enable the subtree functions (for example, for bzr-svn). (Martin Pool)

#### **Features**

- New authentication.conf file holding the password or other credentials for remote servers. This can be used for ssh, sftp, smtp and other supported transports. (Vincent Ladeuil)
- New rich-root and rich-root-pack formats, recording the same data about tree roots that's recorded for all other directories. (Aaron Bentley, #164639)
- pack-0.92 repositories can now be reconciled. (Robert Collins, #154173)
- switch command added for changing the branch a lightweight checkout is associated with and updating the tree to reflect the latest content accordingly. This command was previously part of the BzrTools plug-in. (Ian Clatworthy, Aaron Bentley, David Allouche)
- reconfigure command can now convert branches, trees, or checkouts to lightweight checkouts. (Aaron Bentley)

### Performance

- Commit updates the state of the working tree via a delta rather than supplying entirely new basis trees. For commit of a single specified file this reduces the wall clock time for commit by roughly a 30%. (Robert Collins, Martin Pool)
- Commit with many automatically found deleted paths no longer performs linear scanning for the children of those paths during inventory iteration. This should fix commit performance blowing out when many such paths occur during commit. (Robert Collins, #156491)
- Fetch with pack repositories will no longer read the entire history graph. (Robert Collins, #88319)
- Revert takes out an appropriate lock when reverting to a basis tree, and does not read the basis inventory twice. (Robert Collins)
- Diff does not require an inventory to be generated on dirstate trees. (Aaron Bentley, #149254)
- New annotate merge (-merge-type=weave) implementation is fast on versionedfiles withough cached annotations, e.g. pack-0.92. (Aaron Bentley)

#### **Improvements**

- bzr merge now warns when it encounters a criss-cross merge.
   (Aaron Bentley)
- bzr send now doesn't require the target e-mail address to be specified on the command line if an interactive e-mail client is used. (Lukáš Lalinský)
- bzr tags now prints the revision number for each tag, instead of the revision id, unless –show-ids is passed. In addition, tags can be sorted chronologically instead of lexicographically with – sort=time. (Adeodato Simó, #120231)
- Windows standalone version of bzr is able to load system-wide plugins from "plugins" subdirectory in installation directory. In addition standalone installer write to the registry (HKLMSOFTWAREBazaar) useful info about paths and bzr version. (Alexander Belchenko, #129298)

## Documentation

#### **Bug Fixes**

- A progress bar has been added for knitpack -> knitpack fetching. (Robert Collins, #157789, #159147)
- Branching from a branch via smart server now preserves the repository format. (Andrew Bennetts, #164626)
- commit is now able to invoke an external editor in a non-ascii directory. (Daniel Watkins, #84043)
- Catch connection errors for ftp. (Vincent Ladeuil, #164567)
- check no longer reports spurious unreferenced text versions.
   (Robert Collins, John A Meinel, #162931, #165071)
- Conflicts are now resolved recursively by revert. (Aaron Bentley, #102739)
- Detect invalid transport reuse attempts by catching invalid URLs. (Vincent Ladeuil, #161819)
- Deleting a file without removing it shows a correct diff, not a traceback. (Aaron Bentley)
- Do no use timeout in HttpServer anymore. (Vincent Ladeuil, #158972).
- Don't catch the exceptions related to the http pipeline status before retrying an http request or some programming errors may be masked. (Vincent Ladeuil, #160012)
- Fix bzr rm to not delete modified and ignored files. (Lukáš Lalinský, #172598)
- Fix exception when revisionspec contains merge revisons but log formatter doesn't support merge revisions. (Kent Gibson, #148908)
- Fix exception when ScopeReplacer is assigned to before any members have been retrieved. (Aaron Bentley)
- Fix multiple connections during checkout –lightweight. (Vincent Ladeuil, #159150)
- Fix possible error in insert\_data\_stream when copying between

- pack repositories over bzr+ssh or bzr+http. KnitVersionedFile.get\_data\_stream now makes sure that requested compression parents are sent before any delta hunks that depend on them. (Martin Pool, #164637)
- Fix typo in limiting offsets coalescing for http, leading to whole files being downloaded instead of parts. (Vincent Ladeuil, #165061)
- FTP server errors don't error in the error handling code. (Robert Collins, #161240)
- Give a clearer message when a pull fails because the source needs to be reconciled. (Martin Pool, #164443)
- It is clearer when a plugin cannot be loaded because of its name, and a suggestion for an acceptable name is given. (Daniel Watkins, #103023)
- Leave port as None in transport objects if user doesn't specify a port in urls. (vincent Ladeuil, #150860)
- Make sure Repository.fetch(self) is properly a no-op for all Repository implementations. (John Arbash Meinel, #158333)
- Mark .bzr directories as "hidden" on Windows. (Alexander Belchenko, #71147)
- merge --uncommitted can now operate on a single file. (Aaron Bentley, Lukáš Lalinský, #136890)
- Obsolete packs are now cleaned up by pack and autopack operations. (Robert Collins, #153789)
- Operations pulling data from a smart server where the underlying repositories are not both annotated/both unannotated will now work. (Robert Collins, #165304).
- Reconcile now shows progress bars. (Robert Collins, #159351)
- RemoteBranch was not initializing self.\_revision\_id\_to\_revno\_map properly. (John Arbash Meinel, #162486)
- Removing an already-removed file reports the file does not exist. (Daniel Watkins, #152811)
- Rename on Windows is able to change filename case.

- (Alexander Belchenko, #77740)
- Return error instead of a traceback for bzr log -ro. (Kent Gibson, #133751)
- Return error instead of a traceback when bzr is unable to create symlink on some platforms (e.g. on Windows). (Alexander Belchenko, workaround for #81689)
- Revert doesn't crash when restoring a single file from a deleted directory. (Aaron Bentley)
- Stderr output via logging mechanism now goes through encoded wrapper and no more uses utf-8, but terminal encoding instead. So all unicode strings now should be readable in non-utf-8 terminal. (Alexander Belchenko, #54173)
- The error message when move --after should be used makes how to do so clearer. (Daniel Watkins, #85237)
- Unicode-safe output from bzr info. The output will be encoded using the terminal encoding and unrepresentable characters will be replaced by '?'. (Lukáš Lalinský, #151844)
- Working trees are no longer created when pushing into a local no-trees repo. (Daniel Watkins, #50582)
- Upgrade util/configobj to version 4.4.0. (Vincent Ladeuil, #151208).
- Wrap medusa ftp test server as an FTPServer feature. (Vincent Ladeuil, #157752)

#### **API Breaks**

- osutils.backup\_file is deprecated. Actually it's not used in bzrlib during very long time. (Alexander Belchenko)
- The return value of VersionedFile.iter\_lines\_added\_or\_present\_in\_versions has been changed. Previously it was an iterator of lines, now it is an iterator of (line, version\_id) tuples. This change has been made to aid reconcile and fetch operations. (Robert Collins)
- bzrlib.repository.get\_versioned\_file\_checker is now private.
   (Robert Collins)
- The Repository format registry default has been removed; it was previously obsoleted by the bzrdir format default, which implies a default repository format. (Martin Pool)

#### **Internals**

- Added ContainerSerialiser and ContainerPushParser to bzrlib.pack. These classes provide more convenient APIs for generating and parsing containers from streams rather than from files. (Andrew Bennetts)
- New module <u>lru\_cache</u> providing a cache for use by tasks that need semi-random access to large amounts of data. (John A Meinel)
- InventoryEntry.diff is now deprecated. Please use diff.DiffTree instead.

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

# bzr 0.92

**Released:** 2007-11-05

## Changes

• New uninstaller on Win32. (Alexander Belchenko)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

# bzr 0.92rc1

**Released:** 2007-10-29

# Changes

- bzr now returns exit code 4 if an internal error occurred, and 3 if a normal error occurred. (Martin Pool)
- pull, merge and push will no longer silently correct some repository index errors that occured as a result of the Weave disk format. Instead the reconcile command needs to be run to correct those problems if they exist (and it has been able to fix most such problems since bzr 0.8). Some new problems have been identified during this release and you should run bzr check once on every repository to see if you need to reconcile. If you cannot pull or merge from a remote repository due to mismatched parent errors a symptom of index errors you should simply take a full copy of that remote repository to a clean directory outside any local repositories, then run reconcile on it, and finally pull from it locally. (And naturally email the repositories owner to ask them to upgrade and run reconcile). (Robert Collins)

#### **Features**

- New knitpack-experimental repository format. This is interoperable with the dirstate-tags format but uses a smarter storage design that greatly speeds up many operations, both local and remote. This new format can be used as an option to the init, init-repository and upgrade commands. (Robert Collins)
- For users of bzr-svn (and those testing the prototype subtree support) that wish to try packs, a new knitpack-subtree-experimental format has also been added. This is interoperable with the dirstate-subtrees format. (Robert Collins)
- New reconfigure command. (Aaron Bentley)
- New revert --forget-merges command, which removes the record of a pending merge without affecting the working tree contents. (Martin Pool)
- New bzr\_remote\_path configuration variable allows finer control of remote bzr locations than BZR\_REMOTE\_PATH environment variable. (Aaron Bentley)
- New launchpad-login command to tell Bazaar your Launchpad user ID. This can then be used by other functions of the Launchpad plugin. (James Henstridge)

### Performance

- Commit in quiet mode is now slightly faster as the information to output is no longer calculated. (Ian Clatworthy)
- Commit no longer checks for new text keys during insertion when the revision id was deterministically unique. (Robert Collins)
- Committing a change which is not a merge and does not change the number of files in the tree is faster by utilising the data about whether files are changed to determine if the tree is unchanged rather than recalculating it at the end of the commit process. (Robert Collins)
- Inventory serialisation no longer double-sha's the content. (Robert Collins)
- Knit text reconstruction now avoids making copies of the lines list for interim texts when building a single text. The new apply\_delta method on KnitContent aids this by allowing modification of the revision id such objects represent. (Robert Collins)
- Pack indices are now partially parsed for specific key lookup using a bisection approach. (Robert Collins)
- Partial commits are now approximately 40% faster by walking over the unselected current tree more efficiently. (Robert Collins)
- XML inventory serialisation takes 20% less time while being stricter about the contents. (Robert Collins)
- Graph heads() queries have been fixed to no longer access all history unnecessarily. (Robert Collins)

## **Improvements**

- bzr+https:// smart server across https now supported. (John Ferlito, Martin Pool, #128456)
- Mutt is now a supported mail client; set mail\_client=mutt in your bazaar.conf and send will use mutt. (Keir Mierle)
- New option -c/--change for merge command for cherrypicking changes from one revision. (Alexander Belchenko, #141368)
- Show encodings, locale and list of plugins in the traceback message. (Martin Pool, #63894)
- Experimental directory formats can now be marked with experimental = True during registration. (Ian Clatworthy)

# Documentation

- New Bazaar in Five Minutes guide. (Matthew Revell)
- The hooks reference documentation is now converted to html as expected. (Ian Clatworthy)

### **Bug Fixes**

- Connection error reporting for the smart server has been fixed to display a user friendly message instead of a traceback. (lan Clatworthy, #115601)
- Make sure to use o\_binary when opening files to check their sha1sum. (Alexander Belchenko, John Arbash Meinel, #153493)
- Fix a problem with Win32 handling of the executable bit. (John Arbash Meinel, #149113)
- bzr+ssh:// and sftp:// URLs that do not specify ports explicitly no longer assume that means port 22. This allows people using OpenSSH to override the default port in their ~/.ssh/config if they wish. This fixes a bug introduced in bzr 0.91. (Andrew Bennetts, #146715)
- Commands reporting exceptions can now be profiled and still have their data correctly dumped to a file. For example, a bzr commit with no changes still reports the operation as pointless but doing so no longer throws away the profiling data if this command is run with --lsprof-file callgrind.out.ci say. (lan Clatworthy)
- Fallback to ftp when paramiko is not installed and sftp can't be used for tests/commands so that the test suite is still usable without paramiko. (Vincent Ladeuil, #59150)
- Fix commit ordering in corner case. (Aaron Bentley, #94975)
- Fix long standing bug in partial commit when there are renames left in tree. (Robert Collins, #140419)
- Fix selftest semi-random noise during http related tests. (Vincent Ladeuil, #140614)
- Fix typo in ftp.py making the reconnection fail on temporary errors. (Vincent Ladeuil, #154259)
- Fix failing test by comparing real paths to cover the case where

- the TMPDIR contains a symbolic link. (Vincent Ladeuil, #141382).
- Fix log against smart server branches that don't support tags. (James Westby, #140615)
- Fix pycurl http implementation by defining error codes from pycurl instead of relying on an old curl definition. (Vincent Ladeuil, #147530)
- Fix 'unprintable error' message when displaying BzrCheckError and some other exceptions on Python 2.5. (Martin Pool, #144633)
- Fix Inventory.copy() and add test for it. (Jelmer Vernooij)
- Handles default value for ListOption in cmd\_commit. (Vincent Ladeuil, #140432)
- HttpServer and FtpServer need to be closed properly or a listening socket will remain opened. (Vincent Ladeuil, #140055)
- Monitor the .bzr directory created in the top level test directory to detect leaking tests. (Vincent Ladeuil, #147986)
- The basename, not the full path, is now used when checking whether the profiling dump file begins with callgrind.out or not. This fixes a bug reported by Aaron Bentley on IRC. (Ian Clatworthy)
- Trivial fix for invoking command reconfigure without arguments.
   (Rob Weir, #141629)
- workingTree.rename\_one will now raise an error if normalisation of the new path causes bzr to be unable to access the file. (Robert Collins)
- Correctly detect a NoSuchFile when using a filezilla server. (Gary van der Merwe)

### **API Breaks**

- bzrlib.index.GraphIndex now requires a size parameter to the constructor, for enabling bisection searches. (Robert Collins)
- CommitBuilder.record\_entry\_contents now requires the root entry of a tree be supplied to it, previously failing to do so would trigger a deprecation warning. (Robert Collins)
- KnitVersionedFile.add\* will no longer cache added records even when enable\_cache() has been called - the caching feature is now exclusively for reading existing data. (Robert Collins)
- ReadOnlyLockError is deprecated; LockFailed is usually more appropriate. (Martin Pool)
- Removed bzrlib.transport.TransportLogger please see the new trace+ transport instead. (Robert Collins)
- Removed previously deprecated varargs interface to TestCase.run\_bzr and deprecated methods TestCase.capture and TestCase.run\_bzr\_captured. (Martin Pool)
- Removed previous deprecated basis\_knit parameter to the KnitVersionedFile constructor. (Robert Collins)
- Special purpose method <a href="TestCase.run\_bzr\_decode">TestCase.run\_bzr\_decode</a> is moved to the test\_non\_ascii class that needs it. (Martin Pool)
- The class bzrlib.repofmt.knitrepo.KnitRepository3 has been folded into KnitRepository by parameters to the constructor. (Robert Collins)
- The VersionedFile interface now allows content checks to be bypassed by supplying check\_content=False. This saves nearly 30% of the minimum cost to store a version of a file. (Robert Collins)
- Tree's with bad state such as files with no length or sha will no longer be silently accepted by the repository XML serialiser. To

- serialise inventories without such data, pass working=True to write inventory. (Robert Collins)
- VersionedFile.fix\_parents has been removed as a harmful API. VersionedFile.join will no longer accept different parents on either side of a join - it will either ignore them, or error, depending on the implementation. See notes when upgrading for more information. (Robert Collins)

#### Internals

- bzrlib.transport.Transport.put\_file now returns the number of bytes put by the method call, to allow avoiding stat-after-write or housekeeping in callers. (Robert Collins)
- bzrlib.xml\_serializer.Serializer is now responsible for checking that mandatory attributes are present on serialisation and deserialisation. This fixes some holes in API usage and allows better separation between physical storage and object serialisation. (Robert Collins)
- New class bzrlib.errors.InternalBzrError which is just a convenient shorthand for deriving from BzrError and setting internal\_error = True. (Robert Collins)
- New method bzrlib.mutabletree.update\_to\_one\_parent\_via\_delta for moving the state of a parent tree to a new version via a delta rather than a complete replacement tree. (Robert Collins)
- New method bzrlib.osutils.minimum\_path\_selection useful for removing duplication from user input, when a user mentions both a path and an item contained within that path. (Robert Collins)
- New method bzrlib.repository.Repository.is\_write\_locked useful for determining if a repository is write locked. (Robert Collins)
- New method on bzrlib.tree.Tree path\_content\_summary provides a tuple containing the key information about a path for commit processing to complete. (Robert Collins)
- New method on xml serialisers, write\_inventory\_to\_lines, which matches the API used by knits for adding content. (Robert Collins)
- New module bzrlib.bisect\_multi with generic multiplebisection-at-once logic, currently only available for byte-based

- lookup (bisect\_multi\_bytes). (Robert Collins)
- New helper bzrlib.tuned\_gzip.bytes\_to\_gzip which takes a
  byte string and returns a gzipped version of the same. This is
  used to avoid a bunch of api friction during adding of knit hunks.
  (Robert Collins)
- New parameter on bzrlib.transport.Transport.readv adjust\_for\_latency which changes readv from returning strictly the requested data to inserted return larger ranges and in forward read order to reduce the effect of network latency. (Robert Collins)
- New parameter yield\_parents on Inventory.iter\_entries\_by\_dir which causes the parents of a selected id to be returned recursively, so all the paths from the root down to each element of selected\_file\_ids are returned. (Robert Collins)
- Knit joining has been enhanced to support plain to annotated conversion and annotated to plain conversion. (Ian Clatworthy)
- The CommitBuilder method <a href="record\_entry\_contents">record\_entry\_contents</a> now returns summary information about the effect of the commit on the repository. This tuple contains an inventory delta item if the entry changed from the basis, and a boolean indicating whether a new file graph node was recorded. (Robert Collins)
- The python path used in the Makefile can now be overridden.
   (Andrew Bennetts, Ian Clatworthy)

## **Testing**

- New transport implementation trace+ which is useful for testing, logging activity taken to its \_activity attribute. (Robert Collins)
- When running bzr commands within the test suite, internal exceptions are not caught and reported in the usual way, but rather allowed to propagate up and be visible to the test suite. A new API run\_bzr\_catch\_user\_errors makes this behavior available to other users. (Martin Pool)
- New method TestCase.call\_catch\_warnings for testing methods that raises a Python warning. (Martin Pool)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

# bzr 0.91

**Released:** 2007-09-26

## **Bug Fixes**

- Print a warning instead of aborting the python setup.py install process if building of a C extension is not possible. (Lukáš Lalinský, Alexander Belchenko)
- Fix commit ordering in corner case (Aaron Bentley, #94975)
- Fix "bzr info bzr://host/" and other operations on "bzr://" URLs with an implicit port. We were incorrectly raising PathNotChild due to inconsistent treatment of the "\_port" attribute on the Transport object. (Andrew Bennetts, #133965)
- Make RemoteRepository.sprout cope gracefully with servers that don't support the Repository.tarball request. (Andrew Bennetts)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release

Notes »

previous | next

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

# bzr 0.91rc2

**Released:** 2007-09-11

• Replaced incorrect tarball for previous release; a debug statement was left in bzrlib/remote.py.

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

# bzr 0.91rc1

**Released:** 2007-09-11

## Changes

- The default branch and repository format has changed to dirstate-tags, so tag commands are active by default. This format is compatible with Bazaar 0.15 and later. This incidentally fixes bug #126141. (Martin Pool)
- --quiet or -q is no longer a global option. If present, it must now appear after the command name. Scripts doing things like bzr -q missing need to be rewritten as bzr missing -q. (lan Clatworthy)

#### **Features**

- New option --author in bzr commit to specify the author of the change, if it's different from the committer. bzr log and bzr annotate display the author instead of the committer. (Lukáš Lalinský)
- In addition to global options and command specific options, a set of standard options are now supported. Standard options are legal for all commands. The initial set of standard options are:
  - --help or -h display help message
  - --verbose or -v display additional information
  - --quiet or -q only output warnings and errors.

Unlike global options, standard options can be used in aliases and may have command-specific help. (Ian Clatworthy)

- Verbosity level processing has now been unified. If --verbose or -v is specified on the command line multiple times, the verbosity level is made positive the first time then increased. If --quiet or -q is specified on the command line multiple times, the verbosity level is made negative the first time then decreased. To get the default verbosity level of zero, either specify none of the above, --no-verbose or --no-quiet. Note that most commands currently ignore the magnitude of the verbosity level but do respect quiet vs normal vs verbose when generating output. (lan Clatworthy)
- Branch.hooks now supports pre\_commit hook. The hook's signature is documented in BranchHooks constructor. (Nam T. Nguyen, #102747)

 New Repository.stream\_knit\_data\_for\_revisions request added to the network protocol for greatly reduced roundtrips when retrieving a set of revisions. (Andrew Bennetts)

### **Bug Fixes**

- bzr plugins now lists the version number for each plugin in square brackets after the path. (Robert Collins, #125421)
- Pushing, pulling and branching branches with subtree references was not copying the subtree weave, preventing the file graph from being accessed and causing errors in commits in clones. (Robert Collins)
- Suppress warning "integer argument expected, got float" from Paramiko, which sometimes caused false test failures. (Martin Pool)
- Fix bug in bundle 4 that could cause attempts to write data to wrong versionedfile. (Aaron Bentley)
- Diffs generated using "diff -p" no longer break the patch parser.
   (Aaron Bentley)
- get\_transport treats an empty possible\_transports list the same as a non- empty one. (Aaron Bentley)
- patch verification for merge directives is reactivated, and works with CRLF and CR files. (Aaron Bentley)
- Accept ..as a path in revision specifiers. This fixes for example "r branch:..other-branch" on Windows. (Lukáš Lalinský)
- BZR\_PLUGIN\_PATH may now contain trailing slashes. (Blake Winton, #129299)
- man page no longer lists hidden options (#131667, Aaron Bentley)
- uncommit --help now explains the -r option adequately. (Daniel Watkins, #106726)
- Error messages are now better formatted with parameters (such as filenames) quoted when necessary. This avoids confusion when directory names ending in a '.' at the end of messages were confused with a full stop that may or not have been there. (Daniel Watkins, #129791)

- Fix status FILE -r X..Y. (Lukáš Lalinský)
- If a particular command is an alias, help will show the alias instead of claiming there is no help for said alias. (Daniel Watkins, #133548)
- TreeTransform-based operations, like pull, merge, revert, and branch, now roll back if they encounter an error. (Aaron Bentley, #67699)
- bzr commit now exits cleanly if a character unsupported by the current encoding is used in the commit message. (Daniel Watkins, #116143)
- bzr send uses default values for ranges when only half of an elipsis is specified ("-r..5" or "-r5.."). (#61685, Aaron Bentley)
- Avoid trouble when Windows ssh calls itself 'plink' but no plink binary is present. (Martin Albisetti, #107155)
- bzr remove should remove clean subtrees. Now it will remove (without needing --force) subtrees that contain no files with text changes or modified files. With --force it removes the subtree regardless of text changes or unknown files. Directories with renames in or out (but not changed otherwise) will now be removed without needing --force. Unknown ignored files will be deleted without needing --force. (Marius Kruger, #111665)
- When two plugins conflict, the source of both the losing and now the winning definition is shown. (Konstantin Mikhaylov, #5454)
- When committing to a branch, the location being committed to is displayed. (Daniel Watkins, #52479)
- bzr --version takes care about encoding of stdout, especially when output is redirected. (Alexander Belchenko, #131100)
- Prompt for an ftp password if none is provided. (Vincent Ladeuil, #137044)
- Reuse bound branch associated transport to avoid multiple connections. (Vincent Ladeuil, #128076, #131396)
- Overwrite conflicting tags by push and pull if the --overwrite option is specified. (Lukáš Lalinský, #93947)

- In checkouts, tags are copied into the master branch when created, changed or deleted, and are copied into the checkout when it is updated. (Martin Pool, #93856, #93860)
- Print a warning instead of aborting the python setup.py install process if building of a C extension is not possible. (Lukáš Lalinský, Alexander Belchenko)

### **Improvements**

- Add the option "—show-diff" to the commit command in order to display the diff during the commit log creation. (Goffredo Baroncelli)
- pull and merge are much faster at installing bundle format 4.
   (Aaron Bentley)
- pull -v no longer includes deltas, making it much faster. (Aaron Bentley)
- send now sends the directive as an attachment by default.
   (Aaron Bentley, Lukáš Lalinský, Alexander Belchenko)
- Documentation updates (Martin Albisetti)
- Help on debug flags is now included in help global-options.
   (Daniel Watkins, #124853)
- Parameters passed on the command line are checked to ensure they are supported by the encoding in use. (Daniel Watkins)
- The compression used within the bzr repository has changed from zlib level 9 to the zlib default level. This improves commit performance with only a small increase in space used (and in some cases a reduction in space). (Robert Collins)
- Initial commit no longer SHAs files twice and now reuses the path rather than looking it up again, making it faster. (Ian Clatworthy)
- New option -c/--change for diff and status to show changes in one revision. (Lukáš Lalinský)
- If versioned files match a given ignore pattern, a warning is now given. (Daniel Watkins, #48623)
- bzr status now has -S as a short name for -short and -V as a short name for -versioned. These have been added to assist users migrating from Subversion: bzr status -sv is now like svn status -q. (Daniel Watkins, #115990)
- Added C implementation of PatienceSequenceMatcher, which is

- about 10x faster than the Python version. This speeds up commands that need file diffing, such as bzr commit or bzr diff. (Lukáš Lalinský)
- HACKING has been extended with a large section on core developer tasks. (lan Clatworthy)
- Add branches and standalone-trees as online help topics and include them as Concepts within the User Reference. (Paul Moore, Ian Clatworthy)
- check can detect versionedfile parent references that are inconsistent with revision and inventory info, and reconcile can fix them. These faulty references were generated by 0.8-era releases, so repositories which were manipulated by old bzrs should be checked, and possibly reconciled ASAP. (Aaron Bentley, Andrew Bennetts)

### **API Breaks**

- Branch.append\_revision is removed altogether; please use
   Branch.set\_last\_revision\_info instead. (Martin Pool)
- CommitBuilder now advertises itself as requiring the root entry to be supplied. This only affects foreign repository implementations which reuse CommitBuilder directly and have changed record\_entry\_contents to require that the root not be supplied. This should be precisely zero plugins affected. (Robert Collins)
- The add\_lines methods on versionedFile implementations has changed its return value to include the sha1 and length of the inserted text. This allows the avoidance of double-sha1 calculations during commit. (Robert Collins)
- Transport.should\_cache has been removed. It was not called in the previous release. (Martin Pool)

## **Testing**

- Tests may now raise TestNotApplicable to indicate they shouldn't be run in a particular scenario. (Martin Pool)
- New function multiply\_tests\_from\_modules to give a simpler interface to test parameterization. (Martin Pool, Robert Collins)
- Transport.should\_cache has been removed. It was not called in the previous release. (Martin Pool)
- NULL\_REVISION is returned to indicate the null revision, not None. (Aaron Bentley)
- Use UTF-8 encoded StringIO for log tests to avoid failures on non-ASCII committer names. (Lukáš Lalinský)

### **Internals**

- bzrlib.plugin.all\_plugins has been deprecated in favour of bzrlib.plugin.plugins() which returns Plugln objects that provide useful functionality for determining the path of a plugin, its tests, and its version information. (Robert Collins)
- Add the option user\_encoding to the function 'show\_diff\_trees()' in order to move the user encoding at the UI level. (Goffredo Baroncelli)
- Add the function make\_commit\_message\_template\_encoded()
  and the function edit\_commit\_message\_encoded() which
  handle encoded strings. This is done in order to mix the commit
  messages (which is a unicode string), and the diff which is a raw
  string. (Goffredo Baroncelli)
- CommitBuilder now defaults to using add\_lines\_with\_ghosts, reducing overhead on non-weave repositories which don't require all parents to be present. (Robert Collins)
- Deprecated method find\_previous\_heads on bzrlib.inventory.InventoryEntry. This has been superseded by the use of parent\_candidates and a separate heads check via the repository API. (Robert Collins)
- New trace function mutter\_callsite will print out a subset of the stack to the log, which can be useful for gathering debug details. (Robert Collins)
- bzrlib.pack.ContainerWriter now tracks how many records have been added via a public attribute records\_written. (Robert Collins)
- New method bzrlib.transport.Transport.get\_recommended\_page\_size. This provides a hint to users of transports as to the reasonable minimum data to read. In principle this can take latency and bandwidth into account on a per-connection basis, but for now it

- just has hard coded values based on the url. (e.g. http:// has a large page size, file:// has a small one.) (Robert Collins)
- New method on bzrlib.transport.Transport open\_write\_stream allows incremental addition of data to a file without requiring that all the data be buffered in memory. (Robert Collins)
- New methods on bzrlib.knit.KnitVersionedFile: get\_data\_stream(versions), insert\_data\_stream(stream) and get\_format\_signature(). These provide some infrastructure for efficiently streaming the knit data for a set of versions over the smart protocol.
- Knits with no annotation cache still produce correct annotations.
   (Aaron Bentley)
- Three new methods have been added to bzrlib.trace: set\_verbosity\_level, get\_verbosity\_level and is\_verbose. set\_verbosity\_level expects a numeric value: negative for quiet, zero for normal, positive for verbose. The size of the number can be used to determine just how quiet or verbose the application should be. The existing be\_quiet and is\_quiet routines have been integrated into this new scheme. (lan Clatworthy)
- Options can now be delcared with a custom\_callback parameter.
   If set, this routine is called after the option is processed. This feature is now used by the standard options verbose and quiet so that setting one implicitly resets the other. (Ian Clatworthy)
- Rather than declaring a new option from scratch in order to provide custom help, a centrally registered option can be decorated using the new bzrlib.Option.custom\_help routine. In particular, this routine is useful when declaring better help for the verbose and quiet standard options as the base definition of these is now more complex than before thanks to their use of a custom callback. (Ian Clatworthy)
- Tree.\_iter\_changes(specific\_file=[]) now iterates through no files, instead of iterating through all files. None is used to iterate

through all files. (Aaron Bentley)

• WorkingTree.revert() now accepts None to revert all files. The use of [] to revert all files is deprecated. (Aaron Bentley)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

# bzr 0.90

**Released:** 2007-08-28

### **Improvements**

 Documentation is now organized into multiple directories with a level added for different languages or locales. Added the Mini Tutorial and Quick Start Summary (en) documents from the Wiki, improving the content and readability of the former. Formatted NEWS as Release Notes complete with a Table of Conents, one heading per release. Moved the Developer Guide into the main document catalog and provided a link from the developer document catalog back to the main one. (lan Clatworthy, Sabin lacob, Alexander Belchenko)

### **API Changes**

• The static convenience method <a href="BzrDir.create\_repository">BzrDir.create\_repository</a> is deprecated. Callers should instead create a <a href="BzrDir">BzrDir</a> instance and call <a href="create\_repository">create\_repository</a> on that. (Martin Pool)

# bzr 0.90rc1

**Released:** 2007-08-14

### **Bugfixes**

- bzr init should connect to the remote location one time only.
   We have been connecting several times because we forget to pass around the Transport object. This modifies BzrDir.create\_branch\_convenience, so that we can give it the Transport we already have. (John Arbash Meinel, Vincent Ladeuil, #111702)
- Get rid of sftp connection cache (get rid of the FTP one too).
   (Vincent Ladeuil, #43731)
- bzr branch {local|remote} remote don't try to create a working tree anymore. (Vincent Ladeuil, #112173)
- All identified multiple connections for a single bzr command have been fixed. See bzrlib/tests/commands directory. (Vincent Ladeuil)
- bzr rm now does not insist on --force to delete files that have been renamed but not otherwise modified. (Marius Kruger, #111664)
- bzr selftest --bench no longer emits deprecation warnings (Lukáš Lalinský)
- bzr status now honours FILE parameters for conflict lists (Aaron Bentley, #127606)
- bzr checkout now honours -r when reconstituting a working tree. It also honours -r 0. (Aaron Bentley, #127708)
- bzr add \* no more fails on Windows if working tree contains non-ascii file names. (Kuno Meyer, #127361)
- allow easy\_install bzr runs without fatal errors. (Alexander Belchenko, #125521)
- Graph.\_filter\_candidate\_lca does not raise KeyError if a candidate is eliminated just before it would normally be examined. (Aaron Bentley)
- SMTP connection failures produce a nice message, not a

traceback. (Aaron Bentley)

### **Improvements**

- Don't show "dots" progress indicators when run noninteractively, such as from cron. (Martin Pool)
- info now formats locations more nicely and lists "submit" and "public" branches (Aaron Bentley)
- New pack command that will trigger database compression within the repository (Robert Collins)
- Implement \_KnitIndex.\_load\_data in a pyrex extension. The pyrex version is approximately 2-3x faster at parsing a .kndx file. Which yields a measurable improvement for commands which have to read from the repository, such as a 1s => 0.75s improvement in bzr diff when there are changes to be shown. (John Arbash Meinel)
- Merge is now faster. Depending on the scenario, it can be more than 2x faster. (Aaron Bentley)
- Give a clearer warning, and allow python setup.py install to succeed even if pyrex is not available. (John Arbash Meinel)
- DirState.\_read\_dirblocks now has an optional Pyrex implementation. This improves the speed of any command that has to read the entire DirState. (diff, status, etc, improve by about 10%). bisect\_dirblocks has also been improved, which helps all \_get\_entry type calls (whenever we are searching for a particular entry in the in-memory DirState). (John Arbash Meinel)
- bzr pull and bzr push no longer do a complete walk of the branch revision history for ui display unless -v is supplied. (Robert Collins)
- bzr log -ra..B output shifted to the left margin if the log only contains merge revisions. (Kent Gibson)
- The plugins command is now public with improved help. (Ian

#### Clatworthy)

- New bundle and merge directive formats are faster to generate, and
- Annotate merge now works when there are local changes.
   (Aaron Bentley)
- Commit now only shows the progress in terms of directories instead of entries. (Ian Clatworthy)
- Fix KnitRepository.get\_revision\_graph to not request the graph 2 times. This makes get\_revision\_graph 2x faster. (John Arbash Meinel)
- Fix VersionedFile.get\_graph() to avoid using set.difference\_update(other), which has bad scaling when other is large. This improves VF.get\_graph([version\_id]) for a 12.5k graph from 2.9s down to 200ms. (John Arbash Meinel)
- The --lsprof-file option now generates output for KCacheGrind if the file starts with callgrind.out. This matches the default file filtering done by KCacheGrind's Open Dialog. (lan Clatworthy)
- Fix bzr update to avoid an unnecessary branch.get\_master\_branch call, which avoids 1 extra connection to the remote server. (Partial fix for #128076, John Arbash Meinel)
- Log errors from the smart server in the trace file, to make debugging test failures (and live failures!) easier. (Andrew Bennetts)
- The HTML version of the man page has been superceded by a
  more comprehensive manual called the Bazaar User Reference.
  This manual is completed generated from the online help topics.
  As part of this change, limited reStructuredText is now explicitly
  supported in help topics and command help with 'unnatural'
  markup being removed prior to display by the online help or
  inclusion in the man page. (Ian Clatworthy)
- HTML documentation now use files extension \*.html (Alexander

#### Belchenko)

- The cache of ignore definitions is now cleared in WorkingTree.unlock() so that changes to .bzrignore aren't missed. (#129694, Daniel Watkins)
- bzr selftest --strict fails if there are any missing features or expected test failures. (Daniel Watkins, #111914)
- Link to registration survey added to README. (Ian Clatworthy)
- Windows standalone installer show link to registration survey when installation finished. (Alexander Belchenko)

### Library API Breaks

- KnitIndex.get\_parents now returns tuples. (Robert Collins)
- Ancient unused Repository.text\_store attribute has been removed. (Robert Collins)
- The bzrlib.pack interface has changed to use tuples of bytestrings rather than just bytestrings, making it easier to represent multiple element names. As this interface was not used by any internal facilities since it was introduced in 0.18 no API compatibility is being preserved. The serialised form of these packs is identical with 0.18 when a single element tuple is in use. (Robert Collins)

#### **Internals**

- merge now uses <u>iter\_changes</u> to calculate changes, which makes room for future performance increases. It is also more consistent with other operations that perform comparisons, and reduces reliance on Tree.inventory. (Aaron Bentley)
- Refactoring of transport classes connected to a remote server.
   ConnectedTransport is a new class that serves as a basis for all transports needing to connect to a remote server. transport.split\_url have been deprecated, use the static method on the object instead. URL tests have been refactored too. (Vincent Ladeuil)
- Better connection sharing for ConnectedTransport objects. transport.get\_transport() now accepts a 'possible\_transports' parameter. If a newly requested transport can share a connection with one of the list, it will. (Vincent Ladeuil)
- Most functions now accept bzrlib.revision.NULL\_REVISION to indicate the null revision, and consider using None for this purpose deprecated. (Aaron Bentley)
- New index module with abstract index functionality. This will be used during the planned changes in the repository layer. Currently the index layer provides a graph aware immutable index, a builder for the same index type to allow creating them, and finally a composer for such indices to allow the use of many indices in a single query. The index performance is not optimised, however the API is stable to allow development on top of the index. (Robert Collins)
- bzrlib.dirstate.cmp\_by\_dirs can be used to compare two paths by their directory sections. This is equivalent to comparing path.split('/'), only without having to split the paths. This has a Pyrex implementation available. (John Arbash Meinel)
- New transport decorator 'unlistable+' which disables the list\_dir

- functionality for testing.
- Deprecated change\_entry in transform.py. (lan Clatworthy)
- RevisionTree.get\_weave is now deprecated. Tree.plan\_merge is now used for performing annotate-merge. (Aaron Bentley)
- New EmailMessage class to create email messages. (Adeodato Simó)
- Unused functions on the private interface KnitIndex have been removed. (Robert Collins)
- New knit.KnitGraphIndex which provides a KnitIndex layered on top of a index.GraphIndex. (Robert Collins)
- New knit.KnitVersionedFile.iter\_parents method that allows querying the parents of many knit nodes at once, reducing round trips to the underlying index. (Robert Collins)
- Graph now has an is\_ancestor method, various bits use it.
   (Aaron Bentley)
- The -Dhpss flag now includes timing information. As well as logging when a new connection is opened. (John Arbash Meinel)
- bzrlib.pack.ContainerWriter now returns an offset, length tuple to callers when inserting data, allowing generation of readv style access during pack creation, without needing a separate pass across the output pack to gather such details. (Robert Collins)
- bzrlib.pack.make\_readv\_reader allows readv based access to pack files that are stored on a transport. (Robert Collins)
- New Repository.has\_same\_location method that reports if two repository objects refer to the same repository (although with some risk of false negatives). (Andrew Bennetts)
- InterTree.compare now passes require\_versioned on correctly.
   (Marius Kruger)
- New methods on Repository start\_write\_group,
   commit\_write\_group, abort\_write\_group and is\_in\_write\_group
   which provide a clean hook point for transactional Repositories
  - ones where all the data for a fetch or commit needs to be

- made atomically available in one step. This allows the write lock to remain while making a series of data insertions. (e.g. data conversion). (Robert Collins)
- In bzrlib.knit the internal interface has been altered to use 3-tuples (index, pos, length) rather than two-tuples (pos, length) to describe where data in a knit is, allowing knits to be split into many files. (Robert Collins)
- bzrlib.knit.\_KnitData split into cache management and physical access with two access classes - \_PackAccess and \_KnitAccess defined. The former provides access into a .pack file, and the latter provides the current production repository form of .knit files. (Robert Collins)

### **Testing**

- Remove selftest --clean-output, --numbered-dirs and --keepoutput options, which are obsolete now that tests are done within directories in \$TMPDIR. (Martin Pool)
- The SSH\_AUTH\_SOCK environment variable is now reset to avoid interaction with any running ssh agents. (Jelmer Vernooij, #125955)
- run\_bzr\_subprocess handles parameters the same way as run\_bzr: either a string or a list of strings should be passed as the first parameter. Varargs-style parameters are deprecated. (Aaron Bentley)

# bzr 0.18

**Released:** 2007-07-17

## Bugfixes

• Fix 'bzr add' crash under Win32 (Kuno Meyer)

## bzr 0.18rc1

**Released:** 2007-07-10

### **Bugfixes**

- Do not suppress pipe errors, etc. in non-display commands (Alexander Belchenko, #87178)
- Display a useful error message when the user requests to annotate a file that is not present in the specified revision. (James Westby, #122656)
- Commands that use status flags now have a reference to 'help status-flags'. (Daniel Watkins, #113436)
- Work around python-2.4.1 inhability to correctly parse the authentication header. (Vincent Ladeuil, #121889)
- Use exact encoding for merge directives. (Adeodato Simó, #120591)
- Fix tempfile permissions error in smart server tar bundling under Windows. (Martin \_, #119330)
- Fix detection of directory entries in the inventory. (James Westby)
- Fix handling of http code 400: Bad Request When issuing too many ranges. (Vincent Ladeuil, #115209)
- Issue a CONNECT request when connecting to an https server via a proxy to enable SSL tunneling. (Vincent Ladeuil, #120678)
- Fix bzr log -r to support selecting merge revisions, both individually and as part of revision ranges. (Kent Gibson, #4663)
- Don't leave cruft behind when failing to acquire a lockdir. (Martin Pool, #109169)
- Don't use the '-f' strace option during tests. (Vincent Ladeuil, #102019).
- Warn when setting push\_location to a value that will be masked by locations.conf. (Aaron Bentley, #122286)
- Fix commit ordering in corner case (Aaron Bentley, #94975)
- Make annotate behave in a non-ASCII world (Adeodato Simó).

#### **Improvements**

- The —Isprof-file option now dumps a text rendering of the profiling information if the filename ends in ".txt". It will also convert the profiling information to a format suitable for KCacheGrind if the output filename ends in ".callgrind". Fixes to the Isprofcalltree conversion process by Jean Paul Calderone and Itamar were also merged. See http://ddaa.net/blog/python/Isprof-calltree. (Ian Clatworthy)
- info now defaults to non-verbose mode, displaying only paths and abbreviated format info. info -v displays all the information formerly displayed by info. (Aaron Bentley, Adeodato Simó)
- bzr missing now has better option names --this and --other.
   (Elliot Murphy)
- The internal weave-list command has become versionedfilelist, and now lists knits as well as weaves. (Aaron Bentley)
- Automatic merge base selection uses a faster algorithm that chooses better bases in criss-cross merge situations (Aaron Bentley)
- Progress reporting in commit has been improved. The various logical stages are now reported on as follows, namely:
  - Collecting changes [Entry x/y] Stage n/m
  - Saving data locally Stage n/m
  - Uploading data to master branch Stage n/m
  - Updating the working tree Stage n/m
  - Running post commit hooks Stage n/m

If there is no master branch, the 3rd stage is omitted and the

total number of stages is adjusted accordingly.

Each hook that is run after commit is listed with a name (as hooks can be slow it is useful feedback). (Ian Clatworthy, Robert Collins)

- Various operations that are now faster due to avoiding unnecessary topological sorts. (Aaron Bentley)
- Make merge directives robust against broken bundles. (Aaron Bentley)
- The Isprof filename note is emitted via trace.note(), not standard output. (Aaron Bentley)
- bzrlib now exports explicit API compatibility information to assist library users and plugins. See the bzrlib.api module for details. (Robert Collins)
- Remove unnecessary lock probes when acquiring a lockdir. (Martin Pool)
- bzr --version now shows the location of the bzr log file, which is especially useful on Windows. (Martin Pool)
- -D now supports hooks to get debug tracing of hooks (though its currently minimal in nature). (Robert Collins)
- Long log format reports deltas on merge revisions. (John Arbash Meinel, Kent Gibson)
- Make initial push over ftp more resilient. (John Arbash Meinel)
- Print a summary of changes for update just like pull does. (Daniel Watkins, #113990)
- Add a -Dhpss option to trace smart protocol requests and

responses. (Andrew Bennetts)

### Library API Breaks

- Testing cleanups bzrlib.repository.RepositoryTestProviderAdapter been moved bzrlib.tests.repository\_implementations; bzrlib.repository.InterRepositoryTestProviderAdapter has been moved to bzrlib.tests.interrepository\_implementations; bzrlib.transport.TransportTestProviderAdapter has moved to bzrlib.tests.test\_transport\_implementations. bzrlib.branch.BranchTestProviderAdapter has moved to bzrlib.tests.branch\_implementations. bzrlib.bzrdir.BzrDirTestProviderAdapter has moved to bzrlib.tests.bzrdir\_implementations. bzrlib.versionedfile.InterVersionedFileTestProviderAdapter has moved to bzrlib.tests.interversionedfile\_implementations. bzrlib.store.revision.RevisionStoreTestProviderAdapter moved to bzrlib.tests.revisionstore\_implementations. bzrlib.workingtree.WorkingTreeTestProviderAdapter has moved to bzrlib.tests.workingtree\_implementations. These changes are an API break in the testing infrastructure only. (Robert Collins)
- Relocate TestCaseWithRepository to be more central. (Robert Collins)
- bzrlib.add.smart\_add\_tree will no longer perform glob expansion on win32. Callers of the function should do this and use the new MutableTree.smart\_add method instead. (Robert Collins)
- bzrlib.add.glob\_expand\_for\_win32 is now
   bzrlib.win32utils.glob\_expand. (Robert Collins)

- bzrlib.add.FastPath is now private and moved to bzrlib.mutabletree.\_FastPath. (Robert Collins, Martin Pool)
- LockDir.wait removed. (Martin Pool)
- The smartserver hooks API has changed for the server\_started and server\_stopped hooks. The first parameter is now an iterable of backing URLs rather than a single URL. This is to reflect that many URLs may map to the external URL of the server. E.g. the server interally may have a chrooted URL but also the local file:// URL will be at the same location. (Robert Collins)

#### **Internals**

- New SMTPConnection class to unify email handling. (Adeodato Simó)
- Fix documentation of BzrError. (Adeodato Simó)
- Make BzrBadParameter an internal error. (Adeodato Simó)
- Remove use of 'assert False' to raise an exception unconditionally. (Martin Pool)
- Give a cleaner error when failing to decode knit index entry.
   (Martin Pool)
- TreeConfig would mistakenly search the top level when asked for options from a section. It now respects the section argument and only searches the specified section. (James Westby)
- Improve make api-docs output. (John Arbash Meinel)
- Use os.lstat rather than os.stat for osutils.make\_readonly and osutils.make\_writeable. This makes the difftools plugin more robust when dangling symlinks are found. (Elliot Murphy)
- New -Dlock option to log (to ~/.bzr.log) information on when lockdirs are taken or released. (Martin Pool)
- bzrlib Hooks are now nameable using Hooks.name\_hook. This allows a nicer UI when hooks are running as the current hook can be displayed. (Robert Collins)
- Transport.get has had its interface made more clear for ease of use. Retrieval of a directory must now fail with either 'PathError' at open time, or raise 'ReadError' on a read. (Robert Collins)
- New method \_maybe\_expand\_globs on the command class for dealing with unexpanded glob lists - e.g. on the win32 platform. This was moved from bzrlib.add.\_prepare\_file\_list. (Robert Collins)
- bzrlib.add.smart\_add and bzrlib.add.smart\_add\_tree are now deprecated in favour of MutableTree.smart\_add. (Robert Collins, Martin Pool)

- New method external\_url on Transport for obtaining the url to hand to external processes. (Robert Collins)
- Teach windows installers to build pyrex/C extensions. (Alexander Belchenko)

### **Testing**

- Removed the --keep-output option from selftest and clean up test directories as they're used. This reduces the IO load from running the test suite and cuts the time by about half. (Andrew Bennetts, Martin Pool)
- Add scenarios as a public attribute on the TestAdapter classes to allow modification of the generated scenarios before adaption and easier testing. (Robert Collins)
- New testing support class TestScenarioApplier which multiplies out a single teste by a list of supplied scenarios. (RobertCollins)
- Setting repository\_to\_test\_repository on a repository implementations test will cause it to be called during repository creation, allowing the testing of repository classes which are not based around the Format concept. For example a repository adapter can be tested in this manner, by altering the repository scenarios to include a scenario that sets this attribute during the test parameterisation in bzrlib.tests.repository.repository\_implementations. (Robert Collins)
- Clean up many of the APIs for blackbox testing of Bazaar. The standard interface is now self.run\_bzr. The command to run can be passed as either a list of parameters, a string containing the command line, or (deprecated) varargs parameters. (Martin Pool)
- The base TestCase now isolates tests from -D parameters by clearing debug\_flags and restores it afterwards. (Robert Collins)
- Add a relpath parameter to get\_transport methods in test framework to avoid useless cloning. (Vincent Ladeuil, #110448)

# bzr 0.17

**Released:** 2007-06-18

### **Bugfixes**

- Fix crash of commit due to wrong lookup of filesystem encoding. (Colin Watson, #120647)
- Revert logging just to stderr in commit as broke unicode filenames. (Aaron Bentley, Ian Clatworthy, #120930)

# bzr 0.17rc1

**Released:** 2007-06-12

### **Notes When Upgrading**

• The kind() and is\_executable() APIs on the WorkingTree interface no longer implicitly (read) locks and unlocks the tree. This *might* impact some plug-ins and tools using this part of the API. If you find an issue that may be caused by this change, please let us know, particularly the plug-in/tool maintainer. If encountered, the API fix is to surround kind() and is executable() calls with lock read() and unlock() like so:

```
work_tree.lock_read()
try:
    kind = work_tree.kind(...)
finally:
    work_tree.unlock()
```

#### **Internals**

- Rework of LogFormatter API to provide beginning/end of log hooks and to encapsulate the details of the revision to be logged in a LogRevision object. In long log formats, merge revision ids are only shown when —show-ids is specified, and are labelled "revision-id:", as per mainline revisions, instead of "merged:". (Kent Gibson)
- New BranchBuilder API which allows the construction of particular histories quickly. Useful for testing and potentially other applications too. (Robert Collins)

#### **Improvements**

- There are two new help topics, working-trees and repositories that attempt to explain these concepts. (James Westby, John Arbash Meinel, Aaron Bentley)
- Added bzr log --limit to report a limited number of revisions.
   (Kent Gibson, #3659)
- Revert does not try to preserve file contents that were originally produced by reverting to a historical revision. (Aaron Bentley)
- bzr log --short now includes [merge] for revisions which have more than one parent. This is a small improvement to help understanding what changes have occurred (John Arbash Meinel, #83887)
- TreeTransform avoids many renames when contructing large trees, improving speed. 3.25x speedups have been observed for construction of kernel-sized-trees, and checkouts are 1.28x faster. (Aaron Bentley)
- Commit on large trees is now faster. In my environment, a commit of a small change to the Mozilla tree (55k files) has dropped from 66 seconds to 32 seconds. For a small tree of 600 files, commit of a small change is 33% faster. (Ian Clatworthy)
- New –create-prefix option to bzr init, like for push. (Daniel Watkins, #56322)

### **Bugfixes**

- bzr push should only connect to the remote location one time.
   We have been connecting 3 times because we forget to pass around the Transport object. This adds
   BzrDir.clone\_on\_transport(), so that we can pass in the Transport that we already have. (John Arbash Meinel, #75721)
- DirState.set\_state\_from\_inventory() needs to properly order based on split paths, not just string paths. (John Arbash Meinel, #115947)
- Let TestUIFactoy encode the password prompt with its own stdout. (Vincent Ladeuil, #110204)
- pycurl should take use the range header that takes the range hint into account. (Vincent Ladeuil, #112719)
- WorkingTree4.get\_file\_sha1 no longer raises an exception when invoked on a missing file. (Aaron Bentley, #118186)
- WorkingTree.remove works correctly with tree references, and when pwd is not the tree root. (Aaron Bentley)
- Merge no longer fails when a file is renamed in one tree and deleted in the other. (Aaron Bentley, #110279)
- revision-info now accepts dotted revnos, doesn't require a tree, and defaults to the last revision (Matthew Fuller, #90048)
- Tests no longer fail when BZR\_REMOTE\_PATH is set in the environment. (Daniel Watkins, #111958)
- bzr branch -r revid:foo can be used to branch any revision in your repository. (Previously Branch6 only supported revisions in your mainline). (John Arbash Meinel, #115343)

# bzr 0.16

**Released:** 2007-05-07

### **Bugfixes**

- Handle when you have 2 directories with similar names, but one has a hyphen. ('abc' versus 'abc-2'). The WT4.\_iter\_changes iterator was using direct comparison and 'abc/a' sorts after 'abc-2', but ('abc', 'a') sorts before ('abc-2',). (John Arbash Meinel, #111227)
- Handle when someone renames a file on disk without telling bzr.
   Previously we would report the first file as missing, but not show the new unknown file. (John Arbash Meinel, #111288)
- Avoid error when running hooks after pulling into or pushing from a branch bound to a smartserver branch. (Martin Pool, #111968)

## **Improvements**

- Move developer documentation to doc/developers/. This reduces clutter in the root of the source tree and allows HACKING to be split into multiple files. (Robert Collins, Alexander Belchenko)
- Clean up the WorkingTree4.\_iter\_changes() internal loops as well as DirState.update\_entry(). This optimizes the core logic for bzr diff and bzr status significantly improving the speed of both. (John Arbash Meinel)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release

previous | next

Notes »

# bzr 0.16rc2

**Released:** 2007-04-30

### **Bugfixes**

- Handle the case when you delete a file, and then rename another file on top of it. Also handle the case of bzr rm --keep foo. bzr status should show the removed file and an unknown file in its place. (John Arbash Meinel, #109993)
- Bundles properly read and write revision properties that have an empty value. And when the value is not ASCII. (John Arbash Meinel, #109613)
- Fix the bzr commit message to be in text mode. (Alexander Belchenko, #110901)
- Also handle when you rename a file and create a file where it used to be. (John Arbash Meinel, #110256)
- WorkingTree4.\_iter\_changes should not descend into unversioned directories. (John Arbash Meinel, #110399)

# bzr 0.16rc1

**Released:** 2007-04-26

### **Notes When Upgrading**

• bzr remove and bzr rm will now remove the working file, if it could be recovered again. This has been done for consistency with svn and the unix rm command. The old remove behaviour has been retained in the new option bzr remove --keep, which will just stop versioning the file, but not delete it. bzr remove --force have been added which will always delete the files. bzr remove is also more verbose. (Marius Kruger, #82602)

#### **Improvements**

- Merge directives can now be supplied as input to *merge* and *pull*, like bundles can. (Aaron Bentley)
- Sending the SIGQUIT signal to bzr, which can be done on Unix by pressing Control-Backslash, drops bzr into a debugger. Type 'c' to continue. This can be disabled by setting the environment variable BZR\_SIGQUIT\_PDB=0. (Martin Pool)
- selftest now supports –list-only to list tests instead of running them. (lan Clatworthy)
- selftest now supports –exclude PATTERN (or -x PATTERN) to exclude tests with names that match that regular expression. (lan Clatworthy, #102679)
- selftest now supports –randomize SEED to run tests in a random order. SEED is typically the value 'now' meaning 'use the current time'. (lan Clatworthy, #102686)
- New option --fixes to commit, which stores bug fixing annotations as revision properties. Built-in support for Launchpad, Debian, Trac and Bugzilla bug trackers. (Jonathan Lange, James Henstridge, Robert Collins)
- New API, bzrlib.bugtracker.tracker\_registry, for adding support for other bug trackers to fixes. (Jonathan Lange, James Henstridge, Robert Collins)
- selftest has new short options -f and -1. (Martin Pool)
- bzrlib.tsort.MergeSorter optimizations. Change the inner loop into using local variables instead of going through self.\_var. Improves the time to merge\_sort a 10k revision graph by approximately 40% (~700->400ms). (John Arbash Meinel)
- make docs now creates a man page at man1/bzr.1 fixing bug
   107388. (Robert Collins)
- bzr help now provides cross references to other help topics using the see also facility on command classes. Likewise the

- bzr\_man documentation, and the bzr.1 man page also include this information. (Robert Collins)
- Tags are now included in logs, that use the long log formatter.
   (Erik Bågfors, Alexander Belchenko)
- bzr help provides a clearer message when a help topic cannot be found. (Robert Collins, #107656)
- bzr help now accepts optional prefixes for command help. The help for all commands can now be found at bzr help commands/COMMANDNAME as well as bzr help COMMANDNAME (which only works for commands where the name is not the same as a more general help topic). (Robert Collins)
- bzr help Pluginname will now return the module docstring from the plugin Pluginname. (Robert Collins, #50408)
- New help topic urlspec which lists the availables transports.
   (Goffredo Baroncelli)
- doc/server.txt updated to document the default bzr:// port and also update the blurb about the hpss' current status. (Robert Collins, #107125).
- bzr serve now listens on interface 0.0.0.0 by default, making it serve out to the local LAN (and anyone in the world that can reach the machine running bzr serve. (Robert Collins, #98918)
- A new smart server protocol version has been added. It prefixes requests and responses with an explicit version identifier so that future protocol revisions can be dealt with gracefully. (Andrew Bennetts, Robert Collins)
- The bzr protocol version 2 indicates success or failure in every response without depending on particular commands encoding that consistently, allowing future client refactorings to be much more robust about error handling. (Robert Collins, Martin Pool, Andrew Bennetts)
- The smart protocol over HTTP client has been changed to always post to the same .bzr/smart URL under the original location when it can. This allows HTTP servers to only have to

- pass URLs ending in .bzr/smart to the smart server handler, and not arbitrary .bzr/\*/smart URLs. (Andrew Bennetts)
- digest authentication is now supported for proxies and HTTP by
  the urllib based http implementation. Tested against Apache
  2.0.55 and Squid 2.6.5. Basic and digest authentication are
  handled coherently for HTTP and proxy: if the user is provided
  in the url (bzr command line for HTTP, proxy environment
  variables for proxies), the password is prompted for (only once).
  If the password is provided, it is taken into account. Once the
  first authentication is successful, all further authentication
  roundtrips are avoided by preventively setting the right
  authentication header(s). (Vincent Ladeuil).

#### **Internals**

- bzrlib API compatability with 0.8 has been dropped, cleaning up some code paths. (Robert Collins)
- Change the format of chroot urls so that they can be safely manipulated by generic url utilities without causing the resulting urls to have escaped the chroot. A side effect of this is that creating a chroot requires an explicit action using a ChrootServer. (Robert Collins, Andrew Bennetts)
- Deprecate Branch.get\_root\_id() because branches don't have root ids, rather than fixing bug #96847. (Aaron Bentley)
- WorkingTree.apply\_inventory\_delta provides a better alternative to WorkingTree.\_write\_inventory. (Aaron Bentley)
- Convenience method TestCase.expectFailure ensures that known failures do not silently pass. (Aaron Bentley)
- Transport.local\_abspath now raises NotLocalurl rather than TransportNotPossible. (Martin Pool, Ian Clatworthy)
- New SmartServer hooks facility. There are two initial hooks documented in bzrlib.transport.smart.SmartServerHooks. The two initial hooks allow plugins to execute code upon server startup and shutdown. (Robert Collins).
- SmartServer in standalone mode will now close its listening socket when it stops, rather than waiting for garbage collection. This primarily fixes test suite hangs when a test tries to connect to a shutdown server. It may also help improve behaviour when dealing with a server running on a specific port (rather than dynamically assigned ports). (Robert Collins)
- Move most SmartServer code into a new package, bzrlib/smart. bzrlib/transport/remote.py contains just the Transport classes that used to be in bzrlib/transport/smart.py. (Andrew Bennetts)
- urllib http implementation avoid roundtrips associated with 401 (and 407) errors once the authentication succeeds. (Vincent

- Ladeuil).
- urlib http now supports querying the user for a proxy password if needed. Realm is shown in the prompt for both HTTP and proxy authentication when the user is required to type a password. (Vincent Ladeuil).
- Renamed SmartTransport (and subclasses like SmartTCPTransport) to RemoteTransport (and subclasses to RemoteTCPTransport, etc). This is more consistent with its new home in bzrlib/transport/remote.py, and because it's not really a "smart" transport, just one that does file operations via remote procedure calls. (Andrew Bennetts)
- The lock\_write method of LockableFiles, Repository and Branch now accept a token keyword argument, so that separate instances of those objects can share a lock if it has the right token. (Andrew Bennetts, Robert Collins)
- New method get\_branch\_reference on BzrDir allows the detection of branch references - which the smart server component needs.
- The Repository API make\_working\_trees is now permitted to return False when set\_make\_working\_trees is not implemented previously an unimplemented set\_make\_working\_trees implied the result True from make\_working\_trees. This has been changed to accomodate the smart server, where it does not make sense (at this point) to ever make working trees by default. (Robert Collins)
- Command objects can now declare related help topics by having see also set to a list of related topic. (Robert Collins)
- bzrlib.help now delegates to the Command class for Command specific help. (Robert Collins)
- New class TransportListRegistry, derived from the Registry class, which simplifies tracking the available Transports. (Goffredo Baroncelli)
- New function Branch.get\_revision\_id\_to\_revno\_map which will

return a dictionary mapping revision ids to dotted revnos. Since dotted revnos are defined in the context of the branch tip, it makes sense to generate them from a Branch object. (John Arbash Meinel)

- Fix the 'Unprintable error' message display to use the repr of the exception that prevented printing the error because the str value for it is often not useful in debugging (e.g. KeyError('foo') has a str() of 'foo' but a repr of 'KeyError('foo')' which is much more useful. (Robert Collins)
- urlutils.normalize\_url now unescapes unreserved characters, such as "~". (Andrew Bennetts)

#### **Bugfixes**

- Don't fail bundle selftest if email has 'two' embedded. (lan Clatworthy, #98510)
- Remove --verbose from bzr bundle. It didn't work anyway. (Robert Widhopf-Fenk, #98591)
- Remove --basis from the checkout/branch commands it didn't work properly and is no longer beneficial. (Robert Collins, #53675, #43486)
- Don't produce encoding error when adding duplicate files. (Aaron Bentley)
- Fix bzr log <file> so it only logs the revisions that changed the file, and does it faster. (Kent Gibson, John Arbash Meinel, #51980, #69477)
- Fix InterDirstateTre.\_iter\_changes to handle when we come across an empty versioned directory, which now has files in it. (John Arbash Meinel, #104257)
- Teach common\_ancestor to shortcut when the tip of one branch is inside the ancestry of the other. Saves a lot of graph processing (with an ancestry of 16k revisions, bzr merge ../already-merged changes from 2m10s to 13s). (John Arbash Meinel, #103757)
- Fix <a href="mailto:show\_diff\_trees">show\_diff\_trees</a> to handle the case when a file is modified, and the containing directory is renamed. (The file path is different in this versus base, but it isn't marked as a rename). (John Arbash Meinel, #103870)
- FTP now works even when the FTP server does not support atomic rename. (Aaron Bentley, #89436)
- Correct handling in bundles and merge directives of timezones with that are not an integer number of hours offset from UTC. Always represent the epoch time in UTC to avoid problems with formatting earlier times on win32. (Martin Pool, Alexander Belchenko, John Arbash Meinel)

- Typo in the help for register-branch fixed. (Robert Collins, #96770)
- "dirstate" and "dirstate-tags" formats now produce branches compatible with old versions of bzr. (Aaron Bentley, #107168))
- Handle moving a directory when children have been added, removed, and renamed. (John Arbash Meinel, #105479)
- Don't preventively use basic authentication for proxy before receiving a 407 error. Otherwise people willing to use other authentication schemes may expose their password in the clear (or nearly). This add one roundtrip in case basic authentication should be used, but plug the security hole. (Vincent Ladeuil)
- Handle http and proxy digest authentication. (Vincent Ladeuil, #94034).

### **Testing**

- Added bzrlib.strace.strace which will strace a single callable and return a StraceResult object which contains just the syscalls involved in running it. (Robert Collins)
- New test method reduceLockdirTimeout to drop the default (uicentric) default time down to one suitable for tests. (Andrew Bennetts)
- Add new vfs\_transport\_factory attribute on tests which provides the common vfs backing for both the readonly and readwrite transports. This allows the RemoteObject tests to back onto local disk or memory, and use the existing transport\_server attribute all tests know about to be the smart server transport. This in turn allows tests to differentiate between 'transport to access the branch', and 'transport which is a VFS' which matters in Remote\* tests. (Robert Collins, Andrew Bennetts)
- The make\_branch\_and\_tree method for tests will now create a lightweight checkout for the tree if the vfs\_transport\_factory is not a LocalURLServer. (Robert Collins, Andrew Bennetts)
- Branch implementation tests have been audited to ensure that all urls passed to Branch APIs use proper urls, except when local-disk paths are intended. This is so that tests correctly access the test transport which is often not equivalent to local disk in Remote\* tests. As part of this many tests were adjusted to remove dependencies on local disk access. (Robert Collins, Andrew Bennetts)
- Mark bzrlib.tests and bzrlib.tests.TestUtil as providing assertFOO helper functions by adding a \_\_unittest global attribute. (Robert Collins, Andrew Bennetts, Martin Pool, Jonathan Lange)
- Refactored proxy and authentication handling to simplify the

implementation of new auth schemes for both http and proxy. (Vincent Ladeuil)

# bzr 0.15

**Released:** 2007-04-01

## **Bugfixes**

- Handle incompatible repositories as a user issue when fetching. (Aaron Bentley)
- Don't give a recommendation to upgrade when branching or checking out a branch that contains an old-format working tree. (Martin Pool)

# bzr 0.15rc3

**Released:** 2007-03-26

## Changes

 A warning is now displayed when opening working trees in older formats, to encourage people to upgrade to WorkingTreeFormat4. (Martin Pool)

### **Improvements**

• HTTP redirections are now taken into account when a branch (or a bundle) is accessed for the first time. A message is issued at each redirection to inform the user. In the past, http redirections were silently followed for each request which significantly degraded the performances. The http redirections followed are not anymore by default. instead RedirectRequested exception is raised. For bzrlib users needing http to follow redirections anyway, bzrlib.transport.do\_catching\_redirections provide an easy transition path. (vila)

#### **Internals**

- Added ReadLock.temporary\_write\_lock() to allow upgrading an OS read lock to an OS write lock. Linux can do this without unlocking, Win32 needs to unlock in between. (John Arbash Meinel)
- New parameter recommend\_upgrade to BzrDir.open\_workingtree to silence (when false) warnings about opening old formats. (Martin Pool)
- Fix minor performance regression with bzr-0.15 on pre-dirstate trees. (We were reading the working inventory too many times). (John Arbash Meinel)
- Remove Branch.get\_transaction() in favour of a simple cache
   of revision\_history. Branch subclasses should override
   \_gen\_revision\_history rather than revision\_history to make
   use of this cache, and call \_clear\_revision\_history\_cache and
   \_cache\_revision\_history at appropriate times. (Andrew
   Bennetts)

### **Bugfixes**

- Take smtp\_server from user config into account. (vila, #92195)
- Restore Unicode filename handling for versioned and unversioned files. (John Arbash Meinel, #92608)
- Don't fail during bzr commit if a file is marked removed, and the containing directory is auto-removed. (John Arbash Meinel, #93681)
- bzr status FILENAME failed on Windows because of an uncommon errno. (ERROR\_DIRECTORY == 267 != ENOTDIR).
   (Wouter van Heyst, John Arbash Meinel, #90819)
- bzr checkout source should create a local branch in the same format as source. (John Arbash Meinel, #93854)
- bzr commit with a kind change was failing to update the last-changed-revision for directories. The InventoryDirectory.\_unchanged only looked at the parent\_id and name, ignoring the fact that the kind could have changed, too. (John Arbash Meinel, #90111)
- bzr mv dir/subdir other was incorrectly updating files inside the directory. So that there was a chance it would break commit, etc. (John Arbash Meinel, #94037)
- Correctly handles mutiple permanent http redirections. (vila, #88780)

# bzr 0.15rc2

**Released:** 2007-03-14

## **Notes When Upgrading**

• Release 0.15rc2 of bzr changes the bzr init-repo command to default to --trees instead of --no-trees. Existing shared repositories are not affected.

### **Improvements**

- New merge-directive command to generate machine- and human-readable merge requests. (Aaron Bentley)
- New submit: revision specifier makes it easy to diff against the common ancestor with the submit location (Aaron Bentley)
- Added support for Putty's SSH implementation. (Dmitry Vasiliev)
- Added bzr status --versioned to report only versioned files, not unknowns. (Kent Gibson)
- Merge now autodetects the correct line-ending style for its conflict markers. (Aaron Bentley)

## Internals

 Refactored SSH vendor registration into SSHVendorManager class. (Dmitry Vasiliev)

### **Bugfixes**

- New --numbered-dirs option to bzr selftest to use numbered dirs for TestCaseInTempDir. This is default behavior on Windows. Anyone can force named dirs on Windows with --nonumbered-dirs. (Alexander Belchenko)
- Fix RevisionSpec\_revid to handle the Unicode strings passed in from the command line. (Marien Zwart, #90501)
- Fix TreeTransform.\_iter\_changes when both the source and destination are missing. (Aaron Bentley, #88842)
- Fix commit of merges with symlinks in dirstate trees. (Marien Zwart)
- Switch the bzr init-repo default from —no-trees to —trees. (Wouter van Heyst, #53483)

# bzr 0.15rc1

**Released:** 2007-03-07

### Surprises

- The default disk format has changed. Please run 'bzr upgrade' in your working trees to upgrade. This new default is compatible for network operations, but not for local operations. That is, if you have two versions of bzr installed locally, after upgrading you can only use the bzr 0.15 version. This new default does not enable tags or nested-trees as they are incompatible with bzr versions before 0.15 over the network.
- For users of bzrlib: Two major changes have been made to the working tree api in bzrlib. The first is that many methods and attributes, including the inventory attribute, are no longer valid for use until one of lock\_read/lock\_write/lock\_tree\_write has been called, and become invalid again after unlock is called. This has been done to improve performance and correctness as part of the dirstate development. (Robert Collins, John A Meinel, Martin Pool, and others).
- For users of bzrlib: The attribute 'tree.inventory' should be considered readonly. Previously it was possible to directly alter this attribute, or its contents, and have the tree notice this. This has been made unsupported it may work in some tree formats, but in the newer dirstate format such actions will have no effect and will be ignored, or even cause assertions. All operations possible can still be carried out by a combination of the tree API, and the bzrlib.transform API. (Robert Collins, John A Meinel, Martin Pool, and others).

### **Improvements**

- Support for OS Windows 98. Also .bzr.log on any windows system saved in My Documents folder. (Alexander Belchenko)
- bzr mv enhanced to support already moved files. In the past the mv command would have failed if the source file doesn't exist. In this situation bzr mv would now detect that the file has already moved and update the repository accordingly, if the target file does exist. A new option --after has been added so that if two files already exist, you could notify Bazaar that you have moved a (versioned) file and replaced it with another. Thus in this case bzr move --after will only update the Bazaar identifier. (Steffen Eichenberg, Marius Kruger)
- 1s now works on treeless branches and remote branches.
   (Aaron Bentley)
- bzr help global-options describes the global options. (Aaron Bentley)
- bzr pull --overwrite will now correctly overwrite checkouts.
   (Robert Collins)
- Files are now allowed to change kind (e.g. from file to symlink).
   Supported by commit, revert and status (Aaron Bentley)
- inventory and unknowns hidden in favour of 1s (Aaron Bentley)
- bzr help checkouts descibes what checkouts are and some possible uses of them. (James Westby, Aaron Bentley)
- A new -d option to push, pull and merge overrides the default directory. (Martin Pool)
- Branch format 6: smaller, and potentially faster than format 5.
   Supports append\_history\_only mode, where the log view and revnos do not change, except by being added to. Stores policy settings in ".bzr/branch/branch.conf".
- append\_only branches: Format 6 branches may be configured

- so that log view and revnos are always consistent. Either create the branch using "bzr init —append-revisions-only" or edit the config file as descriped in docs/configuration.txt.
- rebind: Format 6 branches retain the last-used bind location, so if you "bzr unbind", you can "bzr bind" to bind to the previously-selected bind location.
- Builtin tags support, created and deleted by the tag command and stored in the branch. Tags can be accessed with the revisionspec -rtag:, and listed with bzr tags. Tags are not versioned at present. Tags require a network incompatible upgrade. To perform this upgrade, run bzr upgrade --dirstatetags in your branch and repositories. (Martin Pool)
- The bzr:// transport now has a well-known port number, 4155, which it will use by default. (Andrew Bennetts, Martin Pool)
- Bazaar now looks for user-installed plugins before looking for site-wide plugins. (Jonathan Lange)
- bzr resolve now detects and marks resolved text conflicts.
   (Aaron Bentley)

#### Internals

- Internally revision ids and file ids are now passed around as utf-8 bytestrings, rather than treating them as Unicode strings. This has performance benefits for Knits, since we no longer need to decode the revision id for each line of content, nor for each entry in the index. This will also help with the future dirstate format. (John Arbash Meinel)
- Reserved ids (any revision-id ending in a colon) are rejected by versionedfiles, repositories, branches, and working trees (Aaron Bentley)
- Minor performance improvement by not creating a ProgressBar for every KnitIndex we create. (about 90ms for a bzr.dev tree) (John Arbash Meinel)
- New easier to use Branch hooks facility. There are five initial hooks, all documented in bzrlib.branch.BranchHooks.\_\_init\_\_ 'set\_rh', 'post\_push', 'post\_pull', 'post\_commit', 'post\_uncommit'. These hooks fire after the matching operation on a branch has taken place, and were originally added for the branchrss plugin. (Robert Collins)
- New method Branch.push() which should be used when pushing from a branch as it makes performance and policy decisions to match the UI level command push. (Robert Collins).
- Add a new method <u>Tree.revision\_tree</u> which allows access to cached trees for arbitrary revisions. This allows the in development dirstate tree format to provide access to the callers to cached copies of inventory data which are cheaper to access than inventories from the repository. (Robert Collins, Martin Pool)
- New Branch.last\_revision\_info method, this is being done to allow optimization of requests for both the number of revisions and the last revision of a branch with smartservers and

- potentially future branch formats. (Wouter van Heyst, Robert Collins)
- Allow 'import bzrlib.plugins.NAME' to work when the plugin NAME has not yet been loaded by load\_plugins(). This allows plugins to depend on each other for code reuse without requiring users to perform file-renaming gymnastics. (Robert Collins)
- New Repository method 'gather\_stats' for statistic data collection. This is expected to grow to cover a number of related uses mainly related to bzr info. (Robert Collins)
- Log formatters are now managed with a registry. log.register\_formatter continues to work, but callers accessing the FORMATTERS dictionary directly will not.
- Allow a start message to be passed to the edit\_commit\_message function. This will be placed in the message offered to the user for editing above the separator. It allows a template commit message to be used more easily. (James Westby)
- GPGStrategy.sign() will now raise BzrBadParameterUnicode if you pass a Unicode string rather than an 8-bit string. Callers need to be updated to encode first. (John Arbash Meinel)
- Branch.push, pull, merge now return Result objects with information about what happened, rather than a scattering of various methods. These are also passed to the post hooks. (Martin Pool)
- File formats and architecture is in place for managing a forest of trees in bzr, and splitting up existing trees into smaller subtrees, and finally joining trees to make a larger tree. This is the first iteration of this support, and the user-facing aspects still require substantial work. If you wish to experiment with it, use bzr upgrade --dirstate-with-subtree in your working trees and repositories. You can use the hidden commands split and join and to create and manipulate nested trees, but please consider using the nested-trees branch, which contains substantial UI

improvements, instead. http://code.aaronbentley.com/bzr/bzrrepo/nested-trees/ (Aaron Bentley, Martin Pool, Robert Collins).

### **Bugfixes**

- bzr annotate now uses dotted revnos from the viewpoint of the branch, rather than the last changed revision of the file. (John Arbash Meinel, #82158)
- Lock operations no longer hang if they encounter a permission problem. (Aaron Bentley)
- bzr push can resume a push that was canceled before it finished. Also, it can push even if the target directory exists if you supply the --use-existing-dir flag. (John Arbash Meinel, #30576, #45504)
- Fix http proxy authentication when user and an optional password appears in the \*\_proxy vars. (Vincent Ladeuil, #83954).
- bzr log branch/file works for local treeless branches (Aaron Bentley, #84247)
- Fix problem with UNC paths on Windows 98. (Alexander Belchenko, #84728)
- Searching location of CA bundle for PyCurl in env variable (CURL\_CA\_BUNDLE), and on win32 along the PATH. (Alexander Belchenko, #82086)
- bzr init works with unicode argument LOCATION. (Alexander Belchenko, #85599)
- Raise DependencyNotPresent if pycurl do not support https. (Vincent Ladeuil, #85305)
- Invalid proxy env variables should not cause a traceback. (Vincent Ladeuil, #87765)
- Ignore patterns normalised to use '/' path separator. (Kent Gibson, #86451)
- bzr rocks. It sure does! Fix case. (Vincent Ladeuil, #78026)
- Fix bzrtools shelve command for removed lines beginning with "-" (Johan Dahlberg, #75577)

## **Testing**

• New --first option to bzr selftest to run specified tests before the rest of the suite. (Martin Pool)

# bzr 0.14

**Released:** 2007-01-23

## **Improvements**

• bzr help global-options describes the global options. (Aaron Bentley)

### **Bug Fixes**

- Skip documentation generation tests if the tools to do so are not available. Fixes running selftest for installled copies of bzr. (John Arbash Meinel, #80330)
- Fix the code that discovers whether bzr is being run from it's working tree to handle the case when it isn't but the directory it is in is below a repository. (James Westby, #77306)

## bzr 0.14rc1

**Released:** 2007-01-16

### **Improvements**

- New connection: bzr+http:// which supports tunnelling the smart protocol over an HTTP connection. If writing is enabled on the bzr server, then you can write over the http connection. (Andrew Bennetts, John Arbash Meinel)
- Aliases now support quotation marks, so they can contain whitespace (Marius Kruger)
- PyCurlTransport now use a single curl object. By specifying explicitly the 'Range' header, we avoid the need to use two different curl objects (and two connections to the same server). (Vincent Ladeuil)
- bzr commit does not prompt for a message until it is very likely to succeed. (Aaron Bentley)
- bzr conflicts now takes –text to list pathnames of text conflicts (Aaron Bentley)
- Fix iter\_lines\_added\_or\_present\_in\_versions to use a set instead of a list while checking if a revision id was requested. Takes 10s off the fileids\_affected\_by\_revision\_ids time, which is 10s of the bzr branch time. Also improve fileids\_... time by filtering lines with a regex rather than multiple str.find() calls. (saves another 300ms) (John Arbash Meinel)
- Policy can be set for each configuration key. This allows keys to be inherited properly across configuration entries. For example, this should enable you to do:

```
[/home/user/project]
push_location = sftp://host/srv/project/
```

#### push\_location:policy = appendpath

And then a branch like /home/user/project/mybranch should get an automatic push location of sftp://host/srv/project/mybranch. (James Henstridge)

 Added bzr status --short to make status report svn style flags for each file. For example:

```
$ bzr status --short
A foo
A bar
D baz
? wooley
```

- 'bzr selftest -clean-output' allows easily clean temporary tests directories without running tests. (Alexander Belchenko)
- bzr help hidden-commands lists all hidden commands. (Aaron Bentley)
- bzr merge now has an option --pull to fall back to pull if local is fully merged into remote. (Jan Hudec)
- bzr help formats describes available directory formats. (Aaron Bentley)

#### **Internals**

- A few tweaks directly to fileids\_affected\_by\_revision\_ids to help speed up processing, as well allowing to extract unannotated lines. Between the two fileids\_affected\_by\_revision\_ids is improved by approx 10%. (John Arbash Meinel)
- Change Revision serialization to only write out millisecond resolution. Rather than expecting floating point serialization to preserve more resolution than we need. (Henri Weichers, Martin Pool)
- Test suite ends cleanly on Windows. (Vincent Ladeuil)
- When encoding\_type attribute of class Command is equal to 'exact', force sys.stdout to be a binary stream on Windows, and therefore keep exact line-endings (without LF -> CRLF conversion). (Alexander Belchenko)
- Single-letter short options are no longer globally declared. (Martin Pool)
- Before using detected user/terminal encoding bzr should check that Python has corresponding codec. (Alexander Belchenko)
- Formats for end-user selection are provided via a FormatRegistry (Aaron Bentley)

### **Bug Fixes**

- bzr missing --verbose was showing adds/removals in the wrong direction. (John Arbash Meinel)
- bzr annotate now defaults to showing dotted revnos for merged revisions. It cuts them off at a depth of 12 characters, but you can supply --long to see the full number. You can also use --show-ids to display the original revision ids, rather than revision numbers and committer names. (John Arbash Meinel, #75637)
- bzr now supports Win32 UNC path (e.g. \hosт\path. (Alexander Belchenko, #57869)
- Win32-specific: output of cat, bundle and diff commands don't mangle line-endings (Alexander Belchenko, #55276)
- Replace broken fnmatch based ignore pattern matching with custom pattern matcher. (Kent Gibson, Jan Hudec #57637)
- pycurl and urllib can detect short reads at different places. Update the test suite to test more cases. Also detect http error code 416 which was raised for that specific bug. Also enhance the urllib robustness by detecting invalid ranges (and pycurl's one by detecting short reads during the initial GET). (Vincent Ladeuil, #73948)
- The urllib connection sharing interacts badly with urllib2 proxy setting (the connections didn't go thru the proxy anymore). Defining a proper ProxyHandler solves the problem. (Vincent Ladeuil, #74759)
- Use urlutils to generate relative URLs, not osutils (Aaron Bentley, #76229)
- bzr status in a readonly directory should work without giving lots of errors. (John Arbash Meinel, #76299)
- Mention the revisionspec topic for the revision option help. (Wouter van Heyst, #31663)
- Allow plugins import from zip archives. (Alexander Belchenko,

## bzr 0.13

**Released:** 2006-12-05

No changes from 0.13rc

# bzr 0.13rc1

**Released:** 2006-11-27

### **Improvements**

- New command bzr remove-tree allows the removal of the working tree from a branch. (Daniel Silverstone)
- urllib uses shared keep-alive connections, so http operations are substantially faster. (Vincent Ladeuil, #53654)
- bzr export allows an optional branch parameter, to export a bzr tree from some other url. For example: bzr export bzr.tar.gz http://bazaar-vcs.org/bzr/bzr.dev (Daniel Silverstone)
- Added bzr help topics to the bzr help system. This gives a location for general information, outside of a specific command. This includes updates for bzr help revisionspec the first topic included. (Goffredo Baroncelli, John Arbash Meinel, #42714)
- WSGI-compatible HTTP smart server. See doc/http\_smart\_server.txt. (Andrew Bennetts)
- Knit files will now cache full texts only when the size of the deltas is as large as the size of the fulltext. (Or after 200 deltas, whichever comes first). This has the most benefit on large files with small changes, such as the inventory for a large project. (eg For a project with 2500 files, and 7500 revisions, it changes the size of inventory.knit from 11MB to 5.4MB) (John Arbash Meinel)

#### **Internals**

- New -D option given before the command line turns on debugging output for particular areas. -Derror shows tracebacks on all errors. (Martin Pool)
- Clean up bzr selftest --benchmark bundle to correct an import, and remove benchmarks that take longer than 10min to run. (John Arbash Meinel)
- Use time.time() instead of time.clock() to decide on progress throttling. Because time.clock() is actually CPU time, so over a high-latency connection, too many updates get throttled. (John Arbash Meinel)
- MemoryTransport.list\_dir() would strip the first character for files or directories in root directory. (John Arbash Meinel)
- New method get\_branch\_reference on 'BzrDir' allows the detection of branch references - which the smart server component needs.
- New ChrootTransportDecorator, accessible via the chroot+ url prefix. It disallows any access to locations above a set URL. (Andrew Bennetts)

### **Bug Fixes**

- Now \_KnitIndex properly decode revision ids when loading index data. And optimize the knit index parsing code. (Dmitry Vasiliev, John Arbash Meinel)
- bzrlib/bzrdir.py was directly referencing bzrlib.workingtree, without importing it. This prevented bzr upgrade from working unless a plugin already imported bzrlib.workingtree (John Arbash Meinel, #70716)
- Suppress the traceback on invalid URLs (Vincent Ladeuil, #70803).
- Give nicer error message when an http server returns a 403 error code. (Vincent Ladeuil, #57644).
- When a multi-range http GET request fails, try a single range one. If it fails too, forget about ranges. Remember that until the death of the transport and propagates that to the clones. (Vincent Ladeuil, #62276, #62029).
- Handles user/passwords supplied in url from command line (for the urllib implementation). Don't request already known passwords (Vincent Ladeuil, #42383, #44647, #48527)
- \_KnitIndex.add\_versions() dictionary compresses revision ids as they are added. This fixes bug where fetching remote revisions records them as full references rather than integers. (John Arbash Meinel, #64789)
- bzr ignore strips trailing slashes in patterns. Also bzr ignore rejects absolute paths. (Kent Gibson, #4559)
- bzr ignore takes multiple arguments. (Cheuksan Edward Wang, #29488)
- mv correctly handles paths that traverse symlinks. (Aaron Bentley, #66964)
- Give nicer looking error messages when failing to connect over ssh. (John Arbash Meinel, #49172)

- Pushing to a remote branch does not currently update the remote working tree. After a remote push, bzr status and bzr diff on the remote machine now show that the working tree is out of date. (Cheuksan Edward Wang #48136)
- Use patiencediff instead of difflib for determining deltas to insert into knits. This avoids the O(N^3) behavior of difflib. Patience diff should be O(N^2). (Cheuksan Edward Wang, #65714)
- Running bzr log on nonexistent file gives an error instead of the entire log history. (Cheuksan Edward Wang #50793)
- bzr cat can look up contents of removed or renamed files. If the pathname is ambiguous, i.e. the files in the old and new trees have different id's, the default is the file in the new tree. The user can use "—name-from-revision" to select the file in the old tree. (Cheuksan Edward Wang, #30190)

## **Testing**

• TestingHTTPRequestHandler really handles the Range header (previously it was ignoring it and returning the whole file,).

# bzr 0.12

**Released:** 2006-10-30

### **Internals**

• Clean up bzr selftest --benchmark bundle to correct an import, and remove benchmarks that take longer than 10min to run. (John Arbash Meinel)

# bzr 0.12rc1

**Released:** 2006-10-23

## **Improvements**

- bzr log now shows dotted-decimal revision numbers for all revisions, rather than just showing a decimal revision number for revisions on the mainline. These revision numbers are not yet accepted as input into bzr commands such as log, diff etc. (Robert Collins)
- revisions can now be specified using dotted-decimal revision numbers. For instance, bzr diff -r 1.2.1..1.2.3. (Robert Collins)
- bzr help commands output is now shorter (Aaron Bentley)
- bzr now uses lazy importing to reduce the startup time. This has
  a moderate effect on lots of actions, especially ones that have
  little to do. For example bzr rocks time is down to 116ms from
  283ms. (John Arbash Meinel)
- New Registry class to provide name-to-object registry-like support, for example for schemes where plugins can register new classes to do certain tasks (e.g. log formatters). Also provides lazy registration to allow modules to be loaded on request. (John Arbash Meinel, Adeodato Simó)

## **API Incompatability**

• LogFormatter subclasses show now expect the 'revno' parameter to show() to be a string rather than an int. (Robert Collins)

#### **Internals**

- TestCase.run\_bzr, run\_bzr\_captured, and run\_bzr\_subprocess
  can take a working\_dir='foo' parameter, which will change
  directory for the command. (John Arbash Meinel)
- bzrlib.lazy\_regex.lazy\_compile can be used to create a proxy around a regex, which defers compilation until first use. (John Arbash Meinel)
- TestCase.run\_bzr\_subprocess defaults to supplying the --noplugins parameter to ensure test reproducability, and avoid problems with system-wide installed plugins. (John Arbash Meinel)
- Unique tree root ids are now supported. Newly created trees still use the common root id for compatibility with bzr versions before 0.12. (Aaron Bentley)
- WorkingTree.set\_root\_id(None) is now deprecated. Please pass in inventory.ROOT\_ID if you want the default root id value. (Robert Collins, John Arbash Meinel)
- New method workingTree.flush() which will write the current inventory out to disk. At the same read\_working\_inventory will no longer trash the current tree inventory if it has been modified within the current lock, and the will tree automatically now flush() on unlock(). has updated WorkingTree.set\_root\_id() been to take advantage of this functionality. (Robert Collins, John Arbash Meinel)
- bzrlib.tsort.merge\_sorted now accepts generate\_revnos. This
  parameter will cause it to add another column to its output,
  which contains the dotted-decimal revno for each revision, as a
  tuple. (Robert Collins)
- LogFormatter.show\_merge is deprecated in favour of

LogFormatter.show\_merge\_revno. (Robert Collins)

## **Bug Fixes**

- Avoid circular imports by creating a deprecated function for bzrlib.tree.RevisionTree. Callers should have been using bzrlib.revisontree.RevisionTree anyway. (John Arbash Meinel, #66349)
- Don't use socket.MSG\_WAITALL as it doesn't exist on all platforms. (Martin Pool, #66356)
- Don't require content-Type in range responses. Assume they are a single range if content-Type does not exist. (John Arbash Meinel, #62473)
- bzr branch/pull no longer complain about progress bar cleanup when interrupted during fetch. (Aaron Bentley, #54000)
- WorkingTree.set\_parent\_trees() uses the trees to directly write the basis inventory, rather than going through the repository. This allows us to have 1 inventory read, and 2 inventory writes when committing a new tree. (John Arbash Meinel)
- When reverting, files that are not locally modified that do not exist in the target are deleted, not just unversioned (Aaron Bentley)
- When trying to acquire a lock, don't fail immediately. Instead, try
  a few times (up to 1 hour) before timing out. Also, report why the
  lock is unavailable (John Arbash Meinel, #43521, #49556)
- Leave HttpTransportBase daughter classes decides how they implement cloning. (Vincent Ladeuil, #61606)
- diff3 does not indicate conflicts on clean merge. (Aaron Bentley)
- If a commit fails, the commit message is stored in a file at the root of the tree for later commit. (Cheuksan Edward Wang, Stefan Metzmacher, #32054)

## **Testing**

- New test base class TestCaseWithMemoryTransport offers memory-only testing facilities: its not suitable for tests that need to mutate disk state, but most tests should not need that and should be converted to TestCaseWithMemoryTransport. (Robert Collins)
- TestCase.make\_branch\_and\_memory\_tree now takes a format option to set the BzrDir, Repository and Branch formats of the created objects. (Robert Collins, John Arbash Meinel)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

## bzr 0.11

**Released:** 2006-10-02

• Smart server transport test failures on windows fixed. (Lukáš Lalinský).

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

# bzr 0.11rc2

**Released:** 2006-09-27

## **Bug Fixes**

- Test suite hangs on windows fixed. (Andrew Bennets, Alexander Belchenko).
- Commit performance regression fixed. (Aaron Bentley, Robert Collins, John Arbash Meinel).

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

# bzr 0.11rc1

**Released:** 2006-09-25

### **Improvements**

- Knit files now wait to create their contents until the first data is added. The old code used to create an empty .knit and a .kndx with just the header. However, this caused a lot of extra round trips over sftp. This can change the time for bzr push to create a new remote branch from 160s down to 100s. This also affects bzr commit performance when adding new files, bzr commit on a new kernel-like tree drops from 50s down to 40s (John Arbash Meinel, #44692)
- When an entire subtree has been deleted, commit will now report that just the top of the subtree has been deleted, rather than reporting all the individual items. (Robert Collins)
- Commit performs one less XML parse. (Robert Collins)
- bzr checkout now operates on readonly branches as well as readwrite branches. This fixes bug #39542. (Robert Collins)
- bzr bind no longer synchronises history with the master branch.
   Binding should be followed by an update or push to synchronise the two branches. This is closely related to the fix for bug #39542. (Robert Collins)
- bzrlib.lazy\_import.lazy\_import function to create on-demand objects. This allows all imports to stay at the global scope, but modules will not actually be imported if they are not used. (John Arbash Meinel)
- Support bzr:// and bzr+ssh:// urls to work with the new RPC-based transport which will be used with the upcoming high-performance smart server. The new command bzr serve will invoke bzr in server mode, which processes these requests. (Andrew Bennetts, Robert Collins, Martin Pool)
- New command bzr version-info which can be used to get a summary of the current state of the tree. This is especially useful as part of a build commands. See doc/version\_info.txt

for more information (John Arbash Meinel)

## **Bug Fixes**

- 'bzr inventory [FILE...]' allows restricting the file list to a specific set of files. (John Arbash Meinel, #3631)
- Don't abort when annotating empty files (John Arbash Meinel, #56814)
- Add Stanza.to\_unicode() which can be passed to another Stanza when nesting stanzas. Also, add read\_stanza\_unicode to handle when reading a nested Stanza. (John Arbash Meinel)
- Transform.\_set\_mode() needs to stat the right file. (John Arbash Meinel, #56549)
- Raise WeaveFormatError rather than StopIteration when trying to read an empty Weave file. (John Arbash Meinel, #46871)
- Don't access e.code for generic URLErrors, only HTTPErrors have .code. (Vincent Ladeuil, #59835)
- Handle boundary="" lines properly to allow access through a Squid proxy. (John Arbash Meinel, #57723)
- revert now removes newly-added directories (Aaron Bentley, #54172)
- bzr upgrade sftp:// shouldn't fail to upgrade v6 branches if there isn't a working tree. (David Allouche, #40679)
- Give nicer error messages when a user supplies an invalid revision parameter. (John Arbash Meinel, #55420)
- Handle when LANG is not recognized by python. Emit a warning, but just revert to using 'ascii'. (John Arbash Meinel, #35392)
- Don't use preexec\_fn on win32, as it is not supported by subprocess. (John Arbash Meinel)
- Skip specific tests when the dependencies aren't met. This
  includes some setup.py tests when python-dev is not available,
  and some tests that depend on paramiko. (John Arbash Meinel,
  Mattheiu Moy)

- Fallback to Paramiko properly, if no ssh executable exists on the system. (Andrew Bennetts, John Arbash Meinel)
- Branch.bind(other\_branch) no longer takes a write lock on the other branch, and will not push or pull between the two branches. API users will need to perform a push or pull or update operation if they require branch synchronisation to take place. (Robert Collins, #47344)
- When creating a tarball or zipfile export, export unicode names as utf-8 paths. This may not work perfectly on all platforms, but has the best chance of working in the common case. (John Arbash Meinel, #56816)
- When committing, only files that exist in working tree or basis tree may be specified (Aaron Bentley, #50793)

# Portability

• Fixes to run on Python 2.5 (Brian M. Carlson, Martin Pool, Marien Zwart)

#### **Internals**

- TestCaseInTempDir now creates a separate directory for HOME, rather than having HOME set to the same location as the working directory. (John Arbash Meinel)
- run\_bzr\_subprocess() can take an optional env\_changes={}
  parameter, which will update os.environ inside the spawned
  child. It also can take a universal\_newlines=True, which helps
  when checking the output of the command. (John Arbash
  Meinel)
- Refactor SFTP vendors to allow easier re-use when ssh is used.
   (Andrew Bennetts)
- Transport.list\_dir() and Transport.iter\_files\_recursive() should always return urlescaped paths. This is now tested (there were bugs in a few of the transports) (Andrew Bennetts, David Allouche, John Arbash Meinel)
- New utility function symbol\_versioning.deprecation\_string.
   Returns the formatted string for a callable, deprecation format pair. (Robert Collins)
- New TestCase helper applyDeprecated. This allows you to call a callable which is deprecated without it spewing to the screen, just by supplying the deprecation format string issued for it. (Robert Collins)
- Transport.append and Transport.put have been deprecated in favor of .append\_bytes, .append\_file, .put\_bytes, and .put\_file. This removes the ambiguity in what type of object the functions take. Transport.non\_atomic\_put\_{bytes,file} has also been added. Which works similarly to Transport.append() except for SFTP, it doesn't have a round trip when opening the file. Also, it provides functionality for creating a parent directory when trying to create a file, rather than raise NoSuchFile and forcing the caller to repeat their request. (John Arbash Meinel)

- WorkingTree has a new api unversion which allow the unversioning of entries by their file id. (Robert Collins)
- WorkingTree.pending\_merges is deprecated. Please use the get\_parent\_ids (introduced in 0.10) method instead. (Robert Collins)
- WorkingTree has a new lock\_tree\_write method which locks the branch for read rather than write. This is appropriate for actions which only need the branch data for reference rather than mutation. A new decorator needs\_tree\_write\_lock is provided in the workingtree module. Like the needs\_read\_lock and needs\_write\_lock decorators this allows static declaration of the locking requirements of a function to ensure that a lock is taken out for casual scripts. (Robert Collins, #54107)
- All WorkingTree methods which write to the tree, but not to the branch have been converted to use needs\_tree\_write\_lock rather than needs\_write\_lock. Also converted is the revert, conflicts and tree transform modules. This provides a modest performance improvement on metadir style trees, due to the reduce lock-acquisition, and a more significant performance improvement on lightweight checkouts from remote branches, where trivial operations used to pay a significant penalty. It also provides the basis for allowing readonly checkouts. (Robert Collins)
- Special case importing the standard library 'copy' module. This shaves off 40ms of startup time, while retaining compatibility.
   See: bzrlib/inspect\_for\_copy.py for more details. (John Arbash Meinel)
- WorkingTree has a new parent class MutableTree which represents the specialisations of Tree which are able to be altered. (Robert Collins)
- New methods mkdir and put\_file\_bytes\_non\_atomic on MutableTree that mutate the tree and its contents. (Robert Collins)

• Transport behaviour at the root of the URL is now defined and tested. (Andrew Bennetts, Robert Collins)

## **Testing**

- New test helper classs MemoryTree. This is typically accessed via self.make\_branch\_and\_memory\_tree() in test cases. (Robert Collins)
- Add start\_bzr\_subprocess and stop\_bzr\_subprocess to allow test code to continue running concurrently with a subprocess of bzr. (Andrew Bennetts, Robert Collins)
- Add a new method Transport.get\_smart\_client(). This is provided to allow upgrades to a richer interface than the VFS one provided by Transport. (Andrew Bennetts, Martin Pool)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

# bzr 0.10

**Released:** 2006-08-29

## **Improvements**

- 'merge' now takes –uncommitted, to apply uncommitted changes from a tree. (Aaron Bentley)
- 'bzr add —file-ids-from' can be used to specify another path to use for creating file ids, rather than generating all new ones. Internally, the 'action' passed to smart\_add\_tree() can return file\_ids that will be used, rather than having bzrlib generate new ones. (John Arbash Meinel, #55781)
- bzr selftest --benchmark now allows a --cache-dir parameter. This will cache some of the intermediate trees, and decrease the setup time for benchmark tests. (John Arbash Meinel)
- Inverse forms are provided for all boolean options. For example,
   –strict has –no-strict, –no-recurse has –recurse (Aaron Bentley)
- Serialize out Inventories directly, rather than using ElementTree.
   Writing out a kernel sized inventory drops from 2s down to ~350ms. (Robert Collins, John Arbash Meinel)

## **Bug Fixes**

- Help diffutils 2.8.4 get along with binary tests (Marien Zwart: #57614)
- Change LockDir so that if the lock directory doesn't exist when lock\_write() is called, an attempt will be made to create it.
   (John Arbash Meinel, #56974)
- bzr uncommit preserves pending merges. (John Arbash Meinel, #57660)
- Active FTP transport now works as intended. (ghozzy, #56472)
- Really fix mutter() so that it won't ever raise a UnicodeError. It
  means it is possible for ~/.bzr.log to contain non UTF-8
  characters. But it is a debugging log, not a real user file. (John
  Arbash Meinel, #56947, #53880)
- Change Command handle to allow Unicode command and options. At present we cannot register Unicode command names, so we will get BzrCommandError('unknown command'), or BzrCommandError('unknown option') But that is better than a UnicodeError + a traceback. (John Arbash Meinel, #57123)
- Handle TZ=UTC properly when reading/writing revisions. (John Arbash Meinel, #55783, #56290)
- Use GPG\_TTY to allow gpg —cl to work with gpg-agent in a pipeline, (passing text to sign in on stdin). (John Arbash Meinel, #54468)
- External diff does the right thing for binaries even in foreign languages. (John Arbash Meinel, #56307)
- Testament handles more cases when content is unicode.
   Specific bug was in handling of revision properties. (John Arbash Meinel, Holger Krekel, #54723)
- The bzr selftest was failing on installed versions due to a bug in a new test helper. (John Arbash Meinel, Robert Collins, #58057)

#### Internals

- bzrlib.cache\_utf8 contains encode() and decode() functions which can be used to cache the conversion between utf8 and Unicode. Especially helpful for some of the knit annotation code, which has to convert revision ids to utf8 to annotate lines in storage. (John Arbash Meinel)
- setup.py now searches the filesystem to find all packages which need to be installed. This should help make the life of packagers easier. (John Arbash Meinel)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

# bzr 0.9.0

**Released:** 2006-08-11

## Surprises

- The hard-coded built-in ignore rules have been removed. There are now two rulesets which are enforced. A user global one in ~/.bazaar/ignore which will apply to every tree, and the tree specific one '.bzrignore'. ~/.bazaar/ignore will be created if it does not exist, but with a more conservative list than the old default. This fixes bugs with default rules being enforced no matter what. The old list of ignore rules from bzr is available by running 'bzr ignore —old-default-rules'. (Robert Collins, Martin Pool, John Arbash Meinel)
- 'branches.conf' has been changed to 'locations.conf', since it can apply to more locations than just branch locations. (Aaron Bentley)

## **Improvements**

- The revision specifier "revno:" is extended to accept the syntax revno:N:branch. For example, revno:42:http://bazaarvcs.org/bzr/bzr.dev/ means revision 42 in bzr.dev. (Matthieu Moy)
- Tests updates to ensure proper URL handling, UNICODE support, and proper printing when the user's terminal encoding cannot display the path of a file that has been versioned. bzr branch can take a target URL rather than only a local directory. Branch.get\_parent()/set\_parent() now save a relative path if possible, and normalize the parent based on root, allowing access across different transports. (John Arbash Meinel, Wouter van Heyst, Martin Pool) (Malone #48906, #42699, #40675, #5281, #3980, #36363, #43689, #42517, #42514)
- On Unix, detect terminal width using an ioctl not just \$COLUMNS. Use terminal width for single-line logs from bzr log --line and pending-merge display. (Robert Widhopf-Fenk, Gustavo Niemeyer) (Malone #3507)
- On Windows, detect terminal width using GetConsoleScreenBufferInfo. (Alexander Belchenko)
- Speedup improvement for 'date:'-revision search. (Guillaume Pinot).
- Show the correct number of revisions pushed when pushing a new branch. (Robert Collins).
- 'bzr selftest' now shows a progress bar with the number of tests, and progress made. 'make check' shows tests in -v mode, to be more useful for the PQM status window. (Robert Collins). When using a progress bar, failed tests are printed out, rather than being overwritten by the progress bar until the suite finishes. (John Arbash Meinel)
- 'bzr selftest -benchmark' will run a new benchmarking selftest.

'bzr selftest —benchmark —lsprof-timed' will use Isprofile to generate profile data for the individual profiled calls, allowing for fine grained analysis of performance. (Robert Collins, Martin Pool).

- 'bzr commit' shows a progress bar. This is useful for commits over sftp where commit can take an appreciable time. (Robert Collins)
- 'bzr add' is now less verbose in telling you what ignore globs were matched by files being ignored. Instead it just tells you how many were ignored (because you might reasonably be expecting none to be ignored). 'bzr add -v' is unchanged and will report every ignored file. (Robert Collins).
- ftp now has a test server if medusa is installed. As part of testing, ftp support has been improved, including support for supplying a non-standard port. (John Arbash Meinel).
- 'bzr log –line' shows the revision number, and uses only the first line of the log message (#5162, Alexander Belchenko; Matthieu Moy)
- 'bzr status' has had the —all option removed. The 'bzr Is' command should be used to retrieve all versioned files. (Robert Collins)
- 'bzr bundle OTHER/BRANCH' will create a bundle which can be sent over email, and applied on the other end, while maintaining ancestry. This bundle can be applied with either 'bzr merge' or 'bzr pull', the same way you would apply another branch. (John Arbash Meinel, Aaron Bentley)
- 'bzr whoami' can now be used to set your identity from the command line, for a branch or globally. (Robey Pointer)
- 'bzr checkout' now aliased to 'bzr co', and 'bzr annotate' to 'bzr ann'. (Michael Ellerman)
- 'bzr revert DIRECTORY' now reverts the contents of the directory as well. (Aaron Bentley)
- 'bzr get sftp://foo' gives a better error when paramiko is not present. Also updates things like 'http+pycurl://' if pycurl is not

- present. (John Arbash Meinel) (Malone #47821, #52204)
- New env variable BZR\_PROGRESS\_BAR, sets the default progress bar type. Can be set to 'none' or 'dummy' to disable the progress bar, 'dots' or 'tty' to create the respective type. (John Arbash Meinel, #42197, #51107)
- Improve the help text for 'bzr diff' to explain what various options do. (John Arbash Meinel, #6391)
- 'bzr uncommit -r 10' now uncommits revisions 11.. rather than uncommitting revision 10. This makes -r10 more in line with what other commands do. 'bzr uncommit' also now saves the pending merges of the revisions that were removed. So it is safe to uncommit after a merge, fix something, and commit again. (John Arbash Meinel, #32526, #31426)
- 'bzr init' now also works on remote locations. (Wouter van Heyst, #48904)
- HTTP support has been updated. When using pycurl we now support connection keep-alive, which reduces dns requests and round trips. And for both urllib and pycurl we support multi-range requests, which decreases the number of round-trips. Performance results for bzr branch http://bazaarvcs.org/bzr/bzr.dev/ indicate http branching is now 2-3x faster, and bzr pull in an existing branch is as much as 4x faster. (Michael Ellerman, Johan Rydberg, John Arbash Meinel, #46768)
- Performance improvements for sftp. Branching and pulling are now up to 2x faster. Utilize paramiko.readv() support for async requests if it is available (paramiko > 1.6) (John Arbash Meinel)

### **Bug Fixes**

- Fix shadowed definition of TestLocationConfig that caused some tests not to run. (Erik Bågfors, Michael Ellerman, Martin Pool, #32587)
- Fix unnecessary requirement of sign-my-commits that it be run from a working directory. (Martin Pool, Robert Collins)
- 'bzr push location' will only remember the push location if it succeeds in connecting to the remote location. (John Arbash Meinel, #49742)
- 'bzr revert' no longer toggles the executable bit on win32 (John Arbash Meinel, #45010)
- Handle broken pipe under win32 correctly. (John Arbash Meinel)
- sftp tests now work correctly on win32 if you have a newer paramiko (John Arbash Meinel)
- Cleanup win32 test suite, and general cleanup of places where file handles were being held open. (John Arbash Meinel)
- When specifying filenames for 'diff -r x..y', the name of the file in the working directory can be used, even if its name is different in both x and y.
- File-ids containing single- or double-quotes are handled correctly by push. (Aaron Bentley, #52227)
- Normalize unicode filenames to ensure cross-platform consistency. (John Arbash Meinel, #43689)
- The argument parser can now handle '-' as an argument. Currently no code interprets it specially (it is mostly handled as a file named '-'). But plugins, and future operations can use it. (John Arbash meinel, #50984)
- Bundles can properly read binary files with a plain 'r' in them.
   (John Arbash Meinel, #51927)
- Tuning iter\_entries() to be more efficient (John Arbash Meinel, #5444)
- Lots of win32 fixes (the test suite passes again). (John Arbash

- Meinel, #50155)
- Handle openbsd returning None for sys.getfilesystemencoding() (#41183)
- Support ftp APPE (append) to allow Knits to be used over ftp (#42592)
- Removals are only committed if they match the filespec (or if there is no filespec). (#46635, Aaron Bentley)
- smart-add recurses through all supplied directories (John Arbash Meinel, #52578)
- Make the bundle reader extra lines before and after the bundle text. This allows you to parse an email with the bundle inline. (John Arbash Meinel, #49182)
- Change the file id generator to squash a little bit more. Helps when working with long filenames on windows. (Also helps for unicode filenames not generating hidden files). (John Arbash Meinel, #43801)
- Restore terminal mode on C-c while reading sftp password.
   (#48923, Nicholas Allen, Martin Pool)
- Timestamps are rounded to 1ms, and revision entries can be recreated exactly. (John Arbash Meinel, Jamie Wilkinson, #40693)
- Branch.base has changed to a URL, but ~/.bazaar/locations.conf should use local paths, since it is user visible (John Arbash Meinel, #53653)
- bzr status foo when foo was unversioned used to cause a full delta to be generated (John Arbash Meinel, #53638)
- When reading revision properties, an empty value should be considered the empty string, not None (John Arbash Meinel, #47782)
- bzr diff --diff-options can now handle binary files being changed. Also, the output is consistent when –diff-options is not supplied. (John Arbash Meinel, #54651, #52930)
- Use the right suffixes for loading plugins (John Arbash Meinel, #51810)

• Fix Branch.get\_parent() to handle the case when the parent is not accessible (John Arbash Meinel, #52976)

#### Internals

- Combine the ignore rules into a single regex rather than looping over them to reduce the threshold where N^2 behaviour occurs in operations like status. (Jan Hudec, Robert Collins).
- Appending to bzrlib.DEFAULT\_IGNORE is now deprecated.
   Instead, use one of the add functions in bzrlib.ignores. (John Arbash Meinel)
- 'bzr push' should only push the ancestry of the current revision, not all of the history in the repository. This is especially important for shared repositories. (John Arbash Meinel)
- bzrlib.delta.compare\_trees now iterates in alphabetically sorted order, rather than randomly walking the inventories. (John Arbash Meinel)
- Doctests are now run in temporary directories which are cleaned up when they finish, rather than using special ScratchDir/ScratchBranch objects. (Martin Pool)
- Split check into separate methods on the branch and on the repository, so that it can be specialized in ways that are useful or efficient for different formats. (Martin Pool, Robert Collins)
- Deprecate Repository.all\_revision\_ids; most methods don't really need the global revision graph but only that part leading up to a particular revision. (Martin Pool, Robert Collins)
- Add a BzrDirFormat control\_formats list which allows for control formats that do not use '.bzr' to store their data - i.e. '.svn', '.hg' etc. (Robert Collins, Jelmer Vernooij).
- bzrlib.diff.external\_diff can be redirected to any file-like object. Uses subprocess instead of spawnvp. (James Henstridge, John Arbash Meinel, #4047, #48914)
- New command line option '-profile-imports', which will install a custom importer to log time to import modules and regex compilation time to sys.stderr (John Arbash Meinel)

- 'EmptyTree' is now deprecated, please use repository.revision\_tree(None) instead. (Robert Collins)
- "RevisionTree" is now in bzrlib/revisiontree.py. (Robert Collins)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

# bzr 0.8.2

**Released:** 2006-05-17

## **Bug Fixes**

• setup.py failed to install launchpad plugin. (Martin Pool)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

# bzr 0.8.1

**Released:** 2006-05-16

## **Bug Fixes**

- Fix failure to commit a merge in a checkout. (Martin Pool, Robert Collins, Erik Bågfors, #43959)
- Nicer messages from 'commit' in the case of renames, and correct messages when a merge has occured. (Robert Collins, Martin Pool)
- Separate functionality from assert statements as they are skipped in optimized mode of python. Add the same check to pending merges. (Olaf Conradi, #44443)

## Changes

- Do not show the None revision in output of bzr ancestry. (Olaf Conradi)
- Add info on standalone branches without a working tree. (Olaf Conradi, #44155)
- Fix bug in knits when raising InvalidRevisionId. (Olaf Conradi, #44284)

## Changes

 Make editor invocation comply with Debian Policy. First check environment variables VISUAL and EDITOR, then try editor from alternatives system. If that all fails, fall back to the pre-defined list of editors. (Olaf Conradi, #42904)

### **New Features**

• New 'register-branch' command registers a public branch into Launchpad.net, where it can be associated with bugs, etc. (Martin Pool, Bjorn Tillenius, Robert Collins)

## Internals

New public api in InventoryEntry - describe\_change(old, new)
 which provides a human description of the changes between two old and new. (Robert Collins, Martin Pool)

## **Testing**

• Fix test case for bzr info in upgrading a standalone branch to metadir, uses bzrlib api now. (Olaf Conradi)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next Notes »

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

# bzr 0.8

**Released:** 2006-05-08

### **Notes When Upgrading**

Release 0.8 of bzr introduces a new format for history storage, called 'knit', as an evolution of to the 'weave' format used in 0.7. Local and remote operations are faster using knits than weaves. Several operations including 'init', 'init-repo', and 'upgrade' take a –format option that controls this. Branching from an existing branch will keep the same format.

It is possible to merge, pull and push between branches of different formats but this is slower than moving data between homogenous branches. It is therefore recommended (but not required) that you upgrade all branches for a project at the same time. Information on formats is shown by 'bzr info'.

bzr 0.8 now allows creation of 'repositories', which hold the history of files and revisions for several branches. Previously bzr kept all the history for a branch within the .bzr directory at the root of the branch, and this is still the default. To create a repository, use the new 'bzr init-repo' command. Branches exist as directories under the repository and contain just a small amount of information indicating the current revision of the branch.

bzr 0.8 also supports 'checkouts', which are similar to in cvs and subversion. Checkouts are associated with a branch (optionally in a repository), which contains all the historical information. The result is that a checkout can be deleted without losing any already-committed revisions. A new 'update' command is also available.

Repositories and checkouts are not supported with the 0.7 storage format. To use them you must upgrad to either knits, or to the 'metaweave' format, which uses weaves but changes the .bzr directory arrangement.

#### **Improvements**

 sftp paths can now be relative, or local, according to the lftp convention. Paths now take the form:

```
sftp://user:pass@host:port/~/relative/path
or
sftp://user:pass@host:port/absolute/path
```

- The FTP transport now tries to reconnect after a temporary failure. ftp put is made atomic. (Matthieu Moy)
- The FTP transport now maintains a pool of connections, and reuses them to avoid multiple connections to the same host (like sftp did). (Daniel Silverstone)
- The bzr\_man.py file has been removed. To create the man page now, use ./generate\_docs.py man. The new program can also create other files. Run python generate\_docs.py --help for usage information. (Hans Ulrich Niedermann & James Blackwell).
- Man Page now gives full help (James Blackwell). Help also updated to reflect user config now being stored in .bazaar (Hans Ulrich Niedermann)
- It's now possible to set aliases in bazaar.conf (Erik Bågfors)
- Pull now accepts a –revision argument (Erik Bågfors)
- bzr re-sign now allows multiple revisions to be supplied on the command line. You can now use the following command to sign all of your old commits:

```
find .bzr/revision-store// -name my@email-* \
```

- Upgrade can now upgrade over the network. (Robert Collins)
- Two new commands 'bzr checkout' and 'bzr update' allow for CVS/SVN-alike behaviour. By default they will cache history in the checkout, but with –lightweight almost all data is kept in the master branch. (Robert Collins)
- 'revert' unversions newly-versioned files, instead of deleting them.
- 'merge' is more robust. Conflict messages have changed.
- 'merge' and 'revert' no longer clobber existing files that end in '~' or '.moved'.
- Default log format can be set in configuration and plugins can register their own formatters. (Erik Bågfors)
- New 'reconcile' command will check branch consistency and repair indexes that can become out of sync in pre 0.8 formats. (Robert Collins, Daniel Silverstone)
- New 'bzr init –format' and 'bzr upgrade –format' option to control what storage format is created or produced. (Robert Collins, Martin Pool)
- Add parent location to 'bzr info', if there is one. (Olaf Conradi)
- New developer commands 'weave-list' and 'weave-join'. (Martin Pool)
- New 'init-repository' command, plus support for repositories in 'init' and 'branch' (Aaron Bentley, Erik Bågfors, Robert Collins)

- Improve output of 'info' command. Show all relevant locations related to working tree, branch and repository. Use kibibytes for binary quantities. Fix off-by-one error in missing revisions of working tree. Make 'info' work on branches, repositories and remote locations. Show locations relative to the shared repository, if applicable. Show locking status of locations. (Olaf Conradi)
- Diff and merge now safely handle binary files. (Aaron Bentley)
- 'pull' and 'push' now normalise the revision history, so that any two branches with the same tip revision will have the same output from 'log'. (Robert Collins)
- 'merge' accepts –remember option to store parent location, like 'push' and 'pull'. (Olaf Conradi)
- bzr status and diff when files given as arguments do not exist in the relevant trees. (Martin Pool, #3619)
- Add '.hg' to the default ignore list. (Martin Pool)
- 'knit' is now the default disk format. This improves disk performance and utilization, increases incremental pull performance, robustness with SFTP and allows checkouts over SFTP to perform acceptably. The initial Knit code was contributed by Johan Rydberg based on a specification by Martin Pool. (Robert Collins, Aaron Bentley, Johan Rydberg, Martin Pool).
- New tool to generate all-in-one html version of the manual. (Alexander Belchenko)
- Hitting CTRL-C while doing an SFTP push will no longer cause stale locks to be left in the SFTP repository. (Robert Collins, Martin Pool).

- New option 'diff –prefix' to control how files are named in diff output, with shortcuts '-p0' and '-p1' corresponding to the options for GNU patch. (Alexander Belchenko, Goffredo Baroncelli, Martin Pool)
- Add –revision option to 'annotate' command. (Olaf Conradi)
- If bzr shows an unexpected revision-history after pulling (perhaps due to a reweave) it can now be corrected by 'bzr reconcile'. (Robert Collins)

## Changes

- Commit is now verbose by default, and shows changed filenames and the new revision number. (Robert Collins, Martin Pool)
- Unify 'mv', 'move', 'rename'. (Matthew Fuller, #5379)
- 'bzr -h' shows help. (Martin Pool, Ian Bicking, #35940)
- Make 'pull' and 'push' remember location on failure using remember. (Olaf Conradi)
- For compatibility, make old format for using weaves inside metadir available as 'metaweave' format. Rename format 'metadir' to 'default'. Clean up help for option –format in commands 'init', 'init-repo' and 'upgrade'. (Olaf Conradi)

#### **Internals**

- The internal storage of history, and logical branch identity have now been split into Branch, and Repository. The common locking and file management routines are now in bzrlib.lockablefiles. (Aaron Bentley, Robert Collins, Martin Pool)
- Transports can now raise DependencyNotPresent if they need a library which is not installed, and then another implementation will be tried. (Martin Pool)
- Remove obsolete (and no-op) *decode* parameter to *Transport.get*. (Martin Pool)
- Using Tree Transform for merge, revert, tree-building
- WorkingTree.create, WorkingTree.create\_standalone, Branch.initialize are now deprecated. Please see BzrDir.create\_\* for replacement API's. (Robert Collins)
- New BzrDir class represents the .bzr control directory and manages formatting issues. (Robert Collins)
- New repository.InterRepository class encapsulates Repository to Repository actions and allows for clean selection of optimised code paths. (Robert Collins)
- bzrlib.fetch.fetch and bzrlib.fetch.greedy\_fetch are now deprecated, please use branch.fetch or repository.fetch depending on your needs. (Robert Collins)
- deprecated methods now have a is\_deprecated flag on them that can be checked, if you need to determine whether a given callable is deprecated at runtime. (Robert Collins)
- Progress bars are now nested see bzrlib.ui.ui\_factory.nested\_progress\_bar. (Robert Collins, Robey Pointer)
- New API call get\_format\_description() for each type of format.
   (Olaf Conradi)

- Changed branch.set\_parent() to accept None to remove parent. (Olaf Conradi)
- Deprecated BzrError AmbiguousBase. (Olaf Conradi)
- WorkingTree.branch is now a read only property. (Robert Collins)
- bzrlib.ui.text.TextUIFactory now accepts a bar\_type parameter which can be None or a factory that will create a progress bar. This is useful for testing or for overriding the bzrlib.progress heuristic. (Robert Collins)
- New API method get\_physical\_lock\_status() to query locks present on a transport. (Olaf Conradi)
- Repository.reconcile now takes a thorough keyword parameter to allow requesting an indepth reconciliation, rather than just a data-loss check. (Robert Collins)
- bzrlib.ui.ui\_factory protocol now supports get\_boolean to prompt the user for yes/no style input. (Robert Collins)

### **Testing**

- SFTP tests now shortcut the SSH negotiation, reducing test overhead for testing SFTP protocol support. (Robey Pointer)
- Branch formats are now tested once per implementation (see bzrlib. tests.branch\_implementations. This is analagous to the transport interface tests, and has been followed up with working tree, repository and BzrDir tests. (Robert Collins)
- New test base class TestCaseWithTransport provides a transport aware test environment, useful for testing any transport-interface using code. The test suite option –transport controls the transport used by this class (when its not being used as part of implementation contract testing). (Robert Collins)
- Close logging handler on disabling the test log. This will remove the handler from the internal list inside python's logging module, preventing shutdown from closing it twice. (Olaf Conradi)
- Move test case for uncommit to blackbox tests. (Olaf Conradi)
- run\_bzr and run\_bzr\_captured now accept a 'stdin="foo" parameter which will provide String("foo") to the command as its stdin.

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next

Notes »

# bzr 0.7

**Released:** 2006-01-09

## Changes

- .bzrignore is excluded from exports, on the grounds that it's a bzr internal-use file and may not be wanted. (Jamie Wilkinson)
- The "bzr directories" command were removed in favor of the new –kind option to the "bzr inventory" command. To list all versioned directories, now use "bzr inventory –kind directory". (Johan Rydberg)
- Under Windows configuration directory is now %APPDATA%\bazaar\2.0 by default. (John Arbash Meinel)
- The parent of Bzr configuration directory can be set by Bzr\_HOME environment variable. Now the path for it is searched in Bzr\_HOME, then in HOME. Under Windows the order is: Bzr\_HOME, APPDATA (usually points to C:\Documents and Settings\User Name\Application Data), HOME. (John Arbash Meinel)
- Plugins with the same name in different directories in the bzr plugin path are no longer loaded: only the first successfully loaded one is used. (Robert Collins)
- Use systems' external ssh command to open connections if possible. This gives better integration with user settings such as ProxyCommand. (James Henstridge)
- Permissions on files underneath .bzr/ are inherited from the .bzr directory. So for a shared repository, simply doing 'chmod -R g+w .bzr/' will mean that future file will be created with group write permissions.
- configure.in and config.guess are no longer in the builtin default ignore list.
- '.sw[nop]' pattern ignored, to ignore vim swap files for nameless files. (John Arbash Meinel, Martin Pool)

#### **Improvements**

- "bzr INIT dir" now initializes the specified directory, and creates it if it does not exist. (John Arbash Meinel)
- New remerge command (Aaron Bentley)
- Better zsh completion script. (Steve Borho)
- 'bzr diff' now returns 1 when there are changes in the working tree. (Robert Collins)
- 'bzr push' now exists and can push changes to a remote location. This uses the transport infrastructure, and can store the remote location in the ~/.bazaar/branches.conf configuration file. (Robert Collins)
- Test directories are only kept if the test fails and the user requests that they be kept.
- Tweaks to short log printing
- Added branch nicks, new nick command, printing them in log output. (Aaron Bentley)
- If \$BZR\_PDB is set, pop into the debugger when an uncaught exception occurs. (Martin Pool)
- Accept 'bzr resolved' (an alias for 'bzr resolve'), as this is the same as Subversion. (Martin Pool)
- New ftp transport support (on ftplib), for ftp:// and aftp:// URLs. (Daniel Silverstone)
- Commit editor temporary files now start with <a href="bzr\_log">bzr\_log</a>., to allow text editors to match the file name and set up appropriate modes or settings. (Magnus Therning)
- Improved performance when integrating changes from a remote weave. (Goffredo Baroncelli)
- Sftp will attempt to cache the connection, so it is more likely that a connection will be reused, rather than requiring multiple password requests.
- bzr revno now takes an optional argument indicating the branch whose revno should be printed. (Michael Ellerman)

- bzr cat defaults to printing the last version of the file. (Matthieu Moy, #3632)
- New global option 'bzr –lsprof COMMAND' runs bzr under the lsprof profiler. (Denys Duchier)
- Faster commits by reading only the headers of affected weave files. (Denys Duchier)
- 'bzr add' now takes a –dry-run parameter which shows you what would be added, but doesn't actually add anything. (Michael Ellerman)
- 'bzr add' now lists how many files were ignored per glob. add verbose lists the specific files. (Aaron Bentley)
- 'bzr missing' now supports displaying changes in diverged trees and can be limited to show what either end of the comparison is missing. (Aaron Bently, with a little prompting from Daniel Silverstone)

### **Bug Fixes**

- SFTP can walk up to the root path without index errors. (Robert Collins)
- Fix bugs in running bzr with 'python -O'. (Martin Pool)
- Error when run with -OO
- Fix bug in reporting http errors that don't have an http error code. (Martin Pool)
- Handle more cases of pipe errors in display commands
- Change status to 3 for all errors
- Files that are added and unlinked before committing are completely ignored by diff and status
- Stores with some compressed texts and some uncompressed texts are now able to be used. (John A Meinel)
- Fix for bzr pull failing sometimes under windows
- Fix for sftp transport under windows when using interactive auth
- Show files which are both renamed and modified as such in 'bzr status' output. (Daniel Silverstone, #4503)
- Make annotate cope better with revisions committed without a valid email address. (Marien Zwart)
- Fix representation of tab characters in commit messages. (Harald Meland)
- List of plugin directories in BZR\_PLUGIN\_PATH environment variable is now parsed properly under Windows. (Alexander Belchenko)
- Show number of revisions pushed/pulled/merged. (Robey Pointer)
- Keep a cached copy of the basis inventory to speed up operations that need to refer to it. (Johan Rydberg, Martin Pool)
- Fix bugs in bzr status display of non-ascii characters. (Martin Pool)
- Remove Makefile.in from default ignore list. (Tollef Fog Heen, Martin Pool, #6413)

• Fix failure in 'bzr added'. (Nathan McCallum, Martin Pool)

### **Testing**

- Fix selftest asking for passwords when there are no SFTP keys. (Robey Pointer, Jelmer Vernooij)
- Fix selftest run with 'python -O'. (Martin Pool)
- Fix HTTP tests under Windows. (John Arbash Meinel)
- Make tests work even if HOME is not set (Aaron Bentley)
- Updated build\_tree to use fixed line-endings for tests which read the file cotents and compare. Make some tests use this to pass under Windows. (John Arbash Meinel)
- Skip stat and symlink tests under Windows. (Alexander Belchenko)
- Delay in selftest/testhashcash is now issued under win32 and Cygwin. (John Arbash Meinel)
- Use terminal width to align verbose test output. (Martin Pool)
- Blackbox tests are maintained within the bzrlib.tests.blackbox directory. If adding a new test script please add that to bzrlib.tests.blackbox.\_\_init\_\_. (Robert Collins)
- Much better error message if one of the test suites can't be imported. (Martin Pool)
- Make check now runs the test suite twice once with the default locale, and once with all locales forced to C, to expose bugs. This is not trivially done within python, so for now its only triggered by running Make check. Integrators and packagers who wish to check for full platform support should run 'make check' to test the source. (Robert Collins)
- Tests can now run TestSkipped if they can't execute for any reason. (Martin Pool) (NB: TestSkipped should only be raised for correctable reasons - see the wiki spec ImprovingBzrTestSuite).
- Test sftp with relative, absolute-in-homedir and absolute-not-in-homedir paths for the transport tests. Introduce blackbox remote sftp tests that test the same permutations. (Robert Collins, Robey Pointer)

- Transport implementation tests are now independent of the local file system, which allows tests for esoteric transports, and for features not available in the local file system. They also repeat for variations on the URL scheme that can introduce issues in the transport code, see bzrlib.transport.TransportTestProviderAdapter() for this. (Robert Collins).
- TestCase.build\_tree uses the transport interface to build trees, pass in a transport parameter to give it an existing connection. (Robert Collins).

#### Internals

- WorkingTree.pull has been split across Branch and WorkingTree, to allow Branch only pulls. (Robert Collins)
- commands.display\_command now returns the result of the decorated function. (Robert Collins)
- LocationConfig now has a set\_user\_option(key, value) call to save a setting in its matching location section (a new one is created if needed). (Robert Collins)
- Branch has two new methods, get\_push\_location and set\_push\_location to respectively, get and set the push location. (Robert Collins)
- commands.register\_command now takes an optional flag to signal
  that the registrant is planning to decorate an existing command.
  When given multiple plugins registering a command is not an
  error, and the original command class (whether built in or a
  plugin based one) is returned to the caller. There is a new error
  'MustUseDecorated' for signalling when a wrapping command
  should switch to the original version. (Robert Collins)
- Some option parsing errors will raise 'BzrOptionError', allowing granular detection for decorating commands. (Robert Collins).
- Branch.read\_working\_inventory has moved to This necessitated WorkingTree.read\_working\_inventory. changes and of to Branch.get\_root\_id, a move Branch.set\_inventory to WorkingTree as well. To make it clear that WorkingTree obtained a cannot always be Branch.working tree() Will raise errors.NoworkingTree if One

cannot be obtained. (Robert Collins)

- All pending merges operations from Branch are now on WorkingTree. (Robert Collins)
- The follow operations from Branch have moved to WorkingTree:

```
add()
commit()
move()
rename_one()
unknowns()
```

#### (Robert Collins)

- bzrlib.add.smart\_add\_branch is now smart\_add\_tree. (Robert Collins)
- New "rio" serialization format, similar to rfc-822. (Martin Pool)
- Rename selftests to bzrlib.tests.test\_foo. (John A Meinel, Martin Pool)
- bzrlib.plugin.all\_plugins has been changed from an attribute to a query method. (Robert Collins)
- New options to read only the table-of-contents of a weave.
   (Denys Duchier)
- Raise NoSuchFile when someone tries to add a non-existant file. (Michael Ellerman)
- Simplify handling of DivergedBranches in cmd\_pull(). (Michael Ellerman)
- Branch.controlfile\* logic has moved to lockablefiles.LockableFiles, which is exposed as

- Branch().control\_files. Also this has been altered with the controlfile pre/suffix replaced by simple method names like 'get' and 'put'. (Aaron Bentley, Robert Collins).
- Deprecated functions and methods can now be marked as such using the bzrlib.symbol\_versioning module. Marked method have their docstring updated and will issue a DeprecationWarning using the warnings module when they are used. (Robert Collins)
- bzrlib.osutils.safe\_unicode now exists to provide parameter coercion for functions that need unicode strings. (Robert Collins)

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next Notes »

# bzr 0.6

**Released:** 2005-10-28

#### **Improvements**

- pull now takes –verbose to show you what revisions are added or removed (John A Meinel)
- merge now takes a –show-base option to include the base text in conflicts. (Aaron Bentley)
- The config files are now read using ConfigObj, so '=' should be used as a separator, not ':'. (Aaron Bentley)
- New 'bzr commit —strict' option refuses to commit if there are any unknown files in the tree. To commit, make sure all files are either ignored, added, or deleted. (Michael Ellerman)
- The config directory is now ~/.bazaar, and there is a single file ~/.bazaar/bazaar.conf storing email, editor and other preferences. (Robert Collins)
- 'bzr add' no longer takes a –verbose option, and a –quiet option has been added that suppresses all output.
- Improved zsh completion support in contrib/zsh, from Clint Adams.
- Builtin 'bzr annotate' command, by Martin Pool with improvements from Goffredo Baroncelli.
- 'bzr check' now accepts -v for verbose reporting, and checks for ghosts in the branch. (Robert Collins)
- New command 're-sign' which will regenerate the gpg signature for a revision. (Robert Collins)
- If you set check signatures=require for path in a will invoke ~/.bazaar/branches.conf then bzr your gpg\_signing\_command (defaults to gpg) and record a digital signature of your commit. (Robert Collins)
- New sftp transport, based on Paramiko. (Robey Pointer)
- 'bzr pull' now accepts '-clobber' which will discard local changes and make this branch identical to the source branch. (Robert Collins)

- Just give a quieter warning if a plugin can't be loaded, and put the details in .bzr.log. (Martin Pool)
- 'bzr branch' will now set the branch-name to the last component of the output directory, if one was supplied.
- If the option post\_commit is set to one (or more) python function
  names (must be in the bzrlib namespace), then they will be
  invoked after the commit has completed, with the branch and
  revision\_id as parameters. (Robert Collins)
- Merge now has a retcode of 1 when conflicts occur. (Robert Collins)
- -merge-type weave is now supported for file contents. Treeshape changes are still three-way based. (Martin Pool, Aaron Bentley)
- 'bzr check' allows the first revision on revision-history to have parents something that is expected for cheap checkouts, and occurs when conversions from baz do not have all history. (Robert Collins).
- 'bzr merge' can now graft unrelated trees together, if your specify 0 as a base. (Aaron Bentley)
- 'bzr commit branch' and 'bzr commit branch/file1 branch/file2' now work (Aaron Bentley)
- Add '.sconsign\*' to default ignore list. (Alexander Belchenko)
- 'bzr merge –reprocess' minimizes conflicts

#### **Testing**

- The 'bzr selftest –pattern' option for has been removed, now test specifiers on the command line can be simple strings, or regexps, or both. (Robert Collins)
- Passing -v to selftest will now show the time each test took to complete, which will aid in analysing performance regressions and related questions. (Robert Collins)
- 'bzr selftest' runs all tests, even if one fails, unless '-one' is given. (Martin Pool)
- There is a new method for TestCaseInTempDir, assertFileEqual, which will check that a given content is equal to the content of the named file. (Robert Collins)
- Fix test suite's habit of leaving many temporary log files in \$TMPDIR. (Martin Pool)

#### Internals

- New 'testament' command and concept for making gpgsignatures of revisions that are not tied to a particular internal representation. (Martin Pool).
- Per-revision properties ('revprops') as key-value associated strings on each revision created when the revision is committed. Intended mainly for the use of external tools. (Martin Pool).
- Config options have moved from bzrlib.osutils to bzrlib.config. (Robert Collins)
- Improved command line option definitions allowing explanations for individual options, among other things. Contributed by Magnus Therning.
- Config options have moved from bzrlib.osutils to bzrlib.config.
  Configuration is now done via the config.Config interface:
  Depending on whether you have a Branch, a Location or no information available, construct a \*config, and use its signature\_checking, username and user\_email methods. (Robert Collins)
- Plugins are now loaded under bzrlib.plugins, not bzrlib.plugin, and they are made available for other plugins to use. You should not import other plugins during the \_\_init\_\_ of your plugin though, as no ordering is guaranteed, and the plugins directory is not on the python path. (Robert Collins)
- Branch.relpath has been moved to WorkingTree.relpath.
   WorkingTree no no longer takes an inventory, rather it takes an option branch parameter, and if None is given will open the branch at basedir implicitly. (Robert Collins)
- Cleaner exception structure and error reporting. Suggested by Scott James Remnant. (Martin Pool)
- Branch.remove has been moved to WorkingTree, which has also gained lock\_read, lock\_write and unlock methods for

- convenience. (Robert Collins)
- Two decorators, needs\_read\_lock and needs\_write\_lock have been added to the branch module. Use these to cause a function to run in a read or write lock respectively. (Robert Collins)
- Branch.open\_containing now returns a tuple (Branch, relativepath), which allows direct access to the common case of 'get me this file from its branch'. (Robert Collins)
- Transports can register using register\_lazy\_transport, and they will be loaded when first used. (Martin Pool)
- 'pull' has been factored out of the command as WorkingTree.pull(). A new option to WorkingTree.pull has been added, clobber, which will ignore diverged history and pull anyway. (Robert Collins)
- config.Config has a get\_user\_option call that accepts an option name. This will be looked up in branches.conf and bazaar.conf as normal. It is intended that this be used by plugins to support options - options of built in programs should have specific methods on the config. (Robert Collins)
- merge.merge\_inner now has tempdir as an optional parameter.
   (Robert Collins)
- Tree.kind is not recorded at the top level of the hierarchy, as it was missing on EmptyTree, leading to a bug with merge on EmptyTrees. (Robert Collins)
- WorkingTree.\_\_del\_\_ has been removed, it was non deterministic and not doing what it was intended to. See WorkingTree.\_\_init\_\_ for a comment about future directions. (Robert Collins/Martin Pool)
- bzrlib.transport.http has been modified so that only 404 urllib errors are returned as NoSuchFile. Other exceptions will propagate as normal. This allows debuging of actual errors. (Robert Collins)
- bzrlib.transport.Transport now accepts *ONLY* url escaped

- relative paths to apis like 'put', 'get' and 'has'. This is to provide consistent behaviour it operates on url's only. (Robert Collins)
- Transports can register using register\_lazy\_transport, and they will be loaded when first used. (Martin Pool)
- merge\_flex no longer calls conflict\_handler.finalize(), instead that is called by merge\_inner. This is so that the conflict count can be retrieved (and potentially manipulated) before returning to the caller of merge\_inner. Likewise 'merge' now returns the conflict count to the caller. (Robert Collins)
- revision.revision\_graph can handle having only partial history for a revision that is no revisions in the graph with no parents. (Robert Collins).
- New builtins.branch\_files uses the standard file\_list rules to produce a branch and a list of paths, relative to that branch (Aaron Bentley)
- New TestCase.addCleanup facility.
- New bzrlib.version\_info tuple (similar to sys.version\_info), which can be used by programs importing bzrlib.

### **Bug Fixes**

- Better handling of branches in directories with non-ascii names.
   (Joel Rosdahl, Panagiotis Papadakos)
- Upgrades of trees with no commits will not fail due to accessing
   [-1] in the revision-history. (Andres Salomon)

# bzr 0.1.1

**Released:** 2005-10-12

## **Bug Fixes**

- Fix problem in pulling over http from machines that do not allow directories to be listed.
- Avoid harmless warning about invalid hash cache after upgrading branch format.

## Performance

• Avoid some unnecessary http operations in branch and pull.

# bzr 0.1

**Released:** 2005-10-11

### Notes

• 'bzr branch' over http initially gives a very high estimate of completion time but it should fall as the first few revisions are pulled in. branch is still slow on high-latency connections.

#### **Bug Fixes**

- bzr-man.py has been updated to work again. Contributed by Rob Weir.
- Locking is now done with fcntl.lockf which works with NFS file systems. Contributed by Harald Meland.
- When a merge encounters a file that has been deleted on one side and modified on the other, the old contents are written out to foo.BASE and foo.SIDE, where SIDE is this or OTHER. Contributed by Aaron Bentley.
- Export was choosing incorrect file paths for the content of the tarball, this has been fixed by Aaron Bentley.
- Commit will no longer commit without a log message, an error is returned instead. Contributed by Jelmer Vernooij.
- If you commit a specific file in a sub directory, any of its parent directories that are added but not listed will be automatically included. Suggested by Michael Ellerman.
- bzr commit and upgrade did not correctly record new revisions for files with only a change to their executable status. bzr will correct this when it encounters it. Fixed by Robert Collins
- HTTP tests now force off the use of <a href="http\_proxy">http\_proxy</a> for the duration.
   Contributed by Gustavo Niemeyer.
- Fix problems in merging weave-based branches that have different partial views of history.
- Symlink support: working with symlinks when not in the root of a bzr tree was broken, patch from Scott James Remnant.

#### **Improvements**

- 'branch' now accepts a -basis parameter which will take advantage of local history when making a new branch. This allows faster branching of remote branches. Contributed by Aaron Bentley.
- New tree format based on weave files, called version 5. Existing branches can be upgraded to this format using 'bzr upgrade'.
- Symlinks are now versionable. Initial patch by Erik Toubro Nielsen, updated to head by Robert Collins.
- Executable bits are tracked on files. Patch from Gustavo Niemeyer.
- 'bzr status' now shows unknown files inside a selected directory. Patch from Heikki Paajanen.
- Merge conflicts are recorded in .bzr. Two new commands 'conflicts' and 'resolve' have needed added, which list and remove those merge conflicts respectively. A conflicted tree cannot be committed in. Contributed by Aaron Bentley.
- 'rm' is now an alias for 'remove'.
- Stores now split out their content in a single byte prefixed hash, dropping the density of files per directory by 256. Contributed by Gustavo Niemeyer.
- 'bzr diff -r branch:URL' will now perform a diff between two branches. Contributed by Robert Collins.
- 'bzr log' with the default formatter will show merged revisions, indented to the right. Initial implementation contributed by Gustavo Niemeyer, made incremental by Robert Collins.

#### Internals

- Test case failures have the exception printed after the log for your viewing pleasure.
- InventoryEntry is now an abstract base class, use one of the concrete InventoryDirectory etc classes instead.
- Branch raises an UnsupportedFormatError when it detects a bzr branch it cannot understand. This allows for precise handling of such circumstances.
- Remove RevisionReference class; Revision.parent\_ids is now simply a list of their ids and parent\_sha1s is a list of their corresponding sha1s (for old branches only at the moment.)
- New method-object style interface for Commit() and Fetch().
- Renamed Branch.last\_patch() to Branch.last\_revision(),
   since we call them revisions not patches.
- Move copy\_branch to bzrlib.clone.copy\_branch. The destination directory is created if it doesn't exist.
- Inventories now identify the files which were present by giving the revision of that file.
- Inventory and Revision XML contains a version identifier. This
  must be consistent with the overall branch version but allows for
  more flexibility in future upgrades.

## **Testing**

- Removed testsweet module so that tests can be run after bzr installed by 'bzr selftest'.
- 'bzr selftest' command-line arguments can now be partial ids of tests to run, e.g. bzr selftest test\_weave

# bzr 0.0.9

**Released:** 2005-09-23

### **Bug Fixes**

- Fixed "branch -r" option.
- Fix remote access to branches containing non-compressed history. (Robert Collins).
- Better reliability of http server tests. (John Arbash-Meinel)
- Merge graph maximum distance calculation fix. (Aaron Bentley)
- Various minor bug in windows support have been fixed, largely in the test suite. Contributed by Alexander Belchenko.

### **Improvements**

- Status now accepts a -r argument to give status between chosen revisions. Contributed by Heikki Paajanen.
- Revision arguments no longer use +/-/= to control ranges, instead there is a 'before' namespace, which limits the successive namespace. For example '\$ bzr log -r date:yesterday..before:date:today' will select everything from yesterday and before today. Contributed by Robey Pointer
- There is now a bzr.bat file created by distutils when building on Windows. Contributed by Alexander Belchenko.

### Internals

- Removed uuid() as it was unused.
- Improved 'fetch' code for pulling revisions from one branch into another (used by pull, merged, etc.)

# bzr 0.0.8

**Released:** 2005-09-20

#### **Improvements**

- Adding a file whose parent directory is not versioned will implicitly add the parent, and so on up to the root. This means you should never need to explictly add a directory, they'll just get added when you add a file in the directory. Contributed by Michael Ellerman.
- Ignore .ps\_store (contains Mac metadata) by default. (Nir Soffer)
- If you set BZR\_EDITOR in the environment, it is checked in preference to EDITOR and the config file for the interactive commit editing program. Related to this is a bugfix where a missing program set in EDITOR would cause editing to fail, now the fallback program for the operating system is still tried.
- Files that are not directories/symlinks/regular files will no longer cause bzr to fail, it will just ignore them by default. You cannot add them to the tree though they are not versionable.

## Internals

• Refactor xml packing/unpacking.

## **Bug Fixes**

- Fixed 'bzr mv' by Ollie Rutherfurd.
- Fixed strange error when trying to access a nonexistent http branch.
- Make sure that the hashcache gets written out if it can't be read.

# Portability

- Various Windows fixes from Ollie Rutherfurd.
- Quieten warnings about locking; patch from Matt Lavin.

# bzr-0.0.7

**Released:** 2005-09-02

### **New Features**

- bzr shell-complete command contributed by Clint Adams to help with intelligent shell completion.
- New expert command bzr find-merge-base for debugging merges.

### **Enhancements**

- Much better merge support.
- merge3 conflicts are now reported with markers like '<<<<<'
  (seven characters) which is the same as CVS and pleases
  things like emacs smerge.</li>

### **Bug Fixes**

- bzr upgrade no longer fails when trying to fix trees that mention revisions that are not present.
- Fixed bugs in listing plugins from bzr plugins.
- Fix case of \$EDITOR containing options for the editor.
- Fix log -r refusing to show the last revision. (Patch from Goffredo Baroncelli.)

## Changes

- bzr log --show-ids shows the revision ids of all parents.
- Externally provided commands on your \$BZRPATH no longer need to recognize -bzr-usage to work properly, and can just handle -help themselves.

## Library

• Changed trace messages to go through the standard logging framework, so that they can more easily be redirected by libraries.

# bzr-0.0.6

**Released:** 2005-08-18

#### **New Features**

- Python plugins, automatically loaded from the directories on BZR\_PLUGIN\_PATH Or ~/.bzr.conf/plugins by default.
- New 'bzr mkdir' command.
- Commit mesage is fetched from an editor if not given on the command line; patch from Torsten Marek.
- bzr log -m Foo displays commits whose message matches regexp FOO.
- bzr add with no arguments adds everything under the current directory.
- bzr mv does move or rename depending on its arguments, like the Unix command.
- bzr missing command shows a summary of the differences between two trees. (Merged from John Arbash-Meinel.)
- An email address for commits to a particular tree can be specified by putting it into .bzr/email within a branch. (Based on a patch from Heikki Paajanen.)

• Faster working tree operations.

#### Changes

- 3rd-party modules shipped with bzr are copied within the bzrlib python package, so that they can be installed by the setup script without clashing with anything already existing on the system. (Contributed by Gustavo Niemeyer.)
- Moved plugins directory to bzrlib/, so that there's a standard plugin directory which is not only installed with bzr itself but is also available when using bzr from the development tree.
   BZR\_PLUGIN\_PATH and DEFAULT\_PLUGIN\_PATH are then added to the standard plugins directory.
- When exporting to a tarball with bzr export --format tgz, put everything under a top directory rather than dumping it into the current directory. This can be overridden with the --root option. Patch from William Dodé and John Meinel.
- New bzr upgrade command to upgrade the format of a branch, replacing bzr check --update.
- Files within store directories are no longer marked readonly on disk.
- Changed bzr log output to a more compact form suggested by John A Meinel. Old format is available with the --long or -1 option, patched by William Dodé.
- By default the commit command refuses to record a revision with no changes unless the --unchanged option is given.
- The --no-plugins, --profile and --builtin command line options must come before the command name because they affect what commands are available; all other options must come after the command name because their interpretation depends on it.
- branch and clone added as aliases for branch.
- Default log format is back to the long format; the compact one is available with --short.

## **Bug Fixes**

• Fix bugs in committing only selected files or within a subdirectory.

# bzr-0.0.5

**Released:** 2005-06-15

### Changes

- bzr with no command now shows help rather than giving an error. Suggested by Michael Ellerman.
- bzr status output format changed, because svn-style output doesn't really match the model of bzr. Now files are grouped by status and can be shown with their IDs. bzr status --all shows all versioned files and unknown files but not ignored files.
- bzr log runs from most-recent to least-recent, the reverse of the previous order. The previous behaviour can be obtained with the --forward option.
- bzr inventory by default shows only filenames, and also ids if --show-ids is given, in which case the id is the second field.

- New 'bzr whoami –email' option shows only the email component of the user identification, from Jo Vermeulen.
- New bzr ignore PATTERN command.
- Nicer error message for broken pipe, interrupt and similar conditions that don't indicate an internal error.
- Add .\*.sw[nop] .git .\*.tmp \*, v to default ignore patterns.
- Per-branch locks keyed on .bzr/branch-lock, available in either read or write mode.
- New option bzr log --show-ids shows revision and file ids.
- New usage bzr log filename shows only revisions that affected that file.
- Changed format for describing changes in bzr log -v.
- New option bzr commit --file to take a message from a file, suggested by LarstiQ.
- New syntax bzr status [FILE...] contributed by Bartosz Oler. File may be in a branch other than the working directory.
- bzr log and bzr root can be given an http URL instead of a filename.
- Commands can now be defined by external programs or scripts in a directory on \$BZRPATH.
- New "stat cache" avoids reading the contents of files if they haven't changed since the previous time.
- If the Python interpreter is too old, try to find a better one or give an error. Based on a patch from Fredrik Lundh.
- New optional parameter bzr info [BRANCH].
- New form bzr commit SELECTED to commit only selected files.
- New form bzr log -r FROM: TO shows changes in selected range; contributed by John A Meinel.
- New option bzr diff --diff-options 'OPTS' allows passing

- options through to an external GNU diff.
- New option bzr add --no-recurse to add a directory but not their contents.
- bzr --version now shows more information if bzr is being run from a branch.

### **Bug Fixes**

- Fixed diff format so that added and removed files will be handled properly by patch. Fix from Lalo Martins.
- Various fixes for files whose names contain spaces or other metacharacters.

#### **Testing**

- Converted black-box test suites from Bourne shell into Python; now run using ./testbzr. Various structural improvements to the tests.
- testbzr by default runs the version of bzr found in the same directory as the tests, or the one given as the first parameter.
- testbzr also runs the internal tests, so the only command required to check is just ./testbzr.
- testbzr requires python2.4, but can be used to test bzr running under a different version.
- Tests added for many other changes in this release.

#### Internal

- Included ElementTree library upgraded to 1.2.6 by Fredrik Lundh.
- Refactor command functions into Command objects based on HCT by Scott James Remnant.
- Better help messages for many commands.
- Expose <a href="mailto:bzrlib.open\_tracefile">bzrlib.open\_tracefile</a>() to start the tracefile; until this is called trace messages are just discarded.
- New internal function find\_touching\_revisions() and hidden command touching-revisions trace the changes to a given file.
- Simpler and faster compare\_inventories() function.
- bzrlib.open\_tracefile() takes a tracefilename parameter.
- New AtomicFile class.
- New developer commands added, modified.

## Portability

• Cope on Windows on python2.3 by using the weaker random seed. 2.4 is now only recommended.

# bzr-0.0.4

**Released:** 2005-04-22

- 'bzr diff' optionally takes a list of files to diff. Still a bit basic. Patch from QuantumG.
- More default ignore patterns.
- New 'bzr log -verbose' shows a list of files changed in the changeset. Patch from Sebastian Cote.
- Roll over ~/.bzr.log if it gets too large.
- Command abbreviations 'ci', 'st', 'stat', '?' based on a patch by Jason Diamon.
- New 'bzr help commands' based on a patch from Denys Duchier.

#### Changes

- User email is determined by looking at \$BZREMAIL or ~/.bzr.email or \$EMAIL. All are decoded by the locale preferred encoding. If none of these are present user@hostname is used. The host's fully-qualified name is not used because that tends to fail when there are DNS problems.
- New 'bzr whoami' command instead of username user-email.

### **Bug Fixes**

- Make commit safe for hardlinked bzr trees.
- Some Unicode/locale fixes.
- Partial workaround for difflib.unified\_diff not handling trailing newlines properly.

#### Internal

- Allow docstrings for help to be in PEP0257 format. Patch from Matt Brubeck.
- More tests in test.sh.
- Write profile data to a temporary file not into working directory and delete it when done.
- Smaller .bzr.log with process ids.

### **Portability**

- Fix opening of ~/.bzr.log on Windows. Patch from Andrew Bennetts.
- Some improvements in handling paths on Windows, based on a patch from QuantumG.

# bzr-0.0.3

**Released:** 2005-04-06

- New "directories" internal command lists versioned directories in the tree.
- Can now say "bzr commit -help".
- New "rename" command to rename one file to a different name and/or directory.
- New "move" command to move one or more files into a different directory.
- New "renames" command lists files renamed since base revision.
- New cat command contributed by janmar.

### Changes

• .bzr.log is placed in \$HOME (not pwd) and is always written in UTF-8. (Probably not a completely good long-term solution, but will do for now.)

## Portability

• Workaround for difflib bug in Python 2.3 that causes an exception when comparing empty files. Reported by Erik Toubro Nielsen.

### Internal

• Refactored inventory storage to insert a root entry at the top.

# **Testing**

• Start of shell-based black-box testing in test.sh.

# bzr-0.0.2.1

# Portability

• Win32 fixes from Steve Brown.

# bzr-0.0.2

Codename: "black cube" Released: 2005-03-31

- Default ignore list extended (see bzrlib/\_\_init\_\_.py).
- Patterns in .bzrignore are now added to the default ignore list, rather than replacing it.
- Ignore list isn't reread for every file.
- More help topics.
- Reinstate the 'bzr check' command to check invariants of the branch.
- New 'ignored' command lists which files are ignored and why; 'deleted' lists files deleted in the current working tree.
- Performance improvements.
- New global -profile option.
- Ignore patterns like './config.h' now correctly match files in the root directory only.

# bzr-0.0.1

**Released:** 2005-03-26

- More information from info command.
- Can now say "bzr help COMMAND" for more detailed help.
- Less file flushing and faster performance when writing logs and committing to stores.
- More useful verbose output from some commands.

## **Bug Fixes**

• Fix inverted display of 'R' and 'M' during 'commit -v'.

### **Portability**

- Include a subset of ElementTree-1.2.20040618 to make installation easier.
- Fix time.localtime call to work with Python 2.3 (the minimum supported).

# bzr-0.0.0.69

**Released:** 2005-03-22

- First public release.
- Storage of local versions: init, add, remove, rm, info, log, diff, status, etc.

Home | Documentation | Table of Contents (2.2b1) » Bazaar Release previous | next Notes »

## Branches

A branch consists of the state of a project, including all of its history. All branches have a repository associated (which is where the branch history is stored), but multiple branches may share the same repository (a shared repository). Branches can be copied and merged.

In addition, one branch may be bound to another one. Binding to another branch indicates that commits which happen in this branch must also happen in the other branch. Bazaar ensures consistency by not allowing commits when the two branches are out of date. In order for a commit to succeed, it may be necessary to update the current branch using bzr update.

#### Related commands:

```
init Change a directory into a versioned branch.
branch Create a new branch that is a copy of an existing branc
merge Perform a three-way merge.
bind Bind a branch to another one.
```

## Checkouts

Checkouts are source trees that are connected to a branch, so that when you commit in the source tree, the commit goes into that branch. They allow you to use a simpler, more centralized workflow, ignoring some of Bazaar's decentralized features until you want them. Using checkouts with shared repositories is very similar to working with SVN or CVS, but doesn't have the same restrictions. And using checkouts still allows others working on the project to use whatever workflow they like.

A checkout is created with the bzr checkout command (see "help checkout"). You pass it a reference to another branch, and it will create a local copy for you that still contains a reference to the branch you created the checkout from (the master branch). Then if you make any commits they will be made on the other branch first. This creates an instant mirror of your work, or facilitates lockstep development, where each developer is working together, continuously integrating the changes of others.

However the checkout is still a first class branch in Bazaar terms, so that you have the full history locally. As you have a first class branch you can also commit locally if you want, for instance due to the temporary loss af a network connection. Use the —local option to commit to do this. All the local commits will then be made on the master branch the next time you do a non-local commit.

If you are using a checkout from a shared branch you will periodically want to pull in all the changes made by others. This is done using the "update" command. The changes need to be applied before any non-local commit, but Bazaar will tell you if there are any changes and suggest that you use this command when needed.

It is also possible to create a "lightweight" checkout by passing the -

lightweight flag to checkout. A lightweight checkout is even closer to an SVN checkout in that it is not a first class branch, it mainly consists of the working tree. This means that any history operations must query the master branch, which could be slow if a network connection is involved. Also, as you don't have a local branch, then you cannot commit locally.

Lightweight checkouts work best when you have fast reliable access to the master branch. This means that if the master branch is on the same disk or LAN a lightweight checkout will be faster than a heavyweight one for any commands that modify the revision history (as only one copy of the branch needs to be updated). Heavyweight checkouts will generally be faster for any command that uses the history but does not change it, but if the master branch is on the same disk then there won't be a noticeable difference.

Another possible use for a checkout is to use it with a treeless repository containing your branches, where you maintain only one working tree by switching the master branch that the checkout points to when you want to work on a different branch.

Obviously to commit on a checkout you need to be able to write to the master branch. This means that the master branch must be accessible over a writeable protocol, such as sftp://, and that you have write permissions at the other end. Checkouts also work on the local file system, so that all that matters is file permissions.

You can change the master of a checkout by using the "switch" command (see "help switch"). This will change the location that the commits are sent to. The "bind" command can also be used to turn a normal branch into a heavy checkout. If you would like to convert your heavy checkout into a normal branch so that every commit is local, you can use the "unbind" command. To see whether or not a branch is bound or not you can use the "info" command. If the branch is bound it will tell you the location of the bound branch.

#### Related commands:

checkout	Create a checkout. Passlightweight to get a ligh
	checkout
undata	
update	Pull any changes in the master branch in to your ch
commit	Make a commit that is sent to the master branch. If
	a heavy checkout then thelocal option will commi
	checkout without sending the commit to the master
out tob	· · · · · · · · · · · · · · · · · · ·
switch	Change the master branch that the commits in the ch
	be sent to
bind	Turn a standalone branch into a heavy checkout so t
	commits will be sent to the master branch
unbind	Turn a heavy checkout into a standalone branch so t
ulibiliu	
	commits are only made locally
info	Displays whether a branch is bound or unbound. If t
	bound, then it will also display the location of th
.1	
. 1	Þ

# **Content Filters**

### **Content formats**

Bazaar's content filtering allows you to store files in a different format from the copy in your working tree. This lets you, or your codevelopers, use Windows development tools that expect CRLF files on projects that use other line-ending conventions. Among other things, content filters also let Linux developers more easily work on projects using Windows line-ending conventions, keyword expansion/compression, and trailing spaces on lines in text files to be implicitly stripped when committed.

To generalize, there are two content formats supported by Bazaar:

- a canonical format how files are stored internally
- a convenience format how files are created in a working tree.

#### Format conversion

The conversion between these formats is done by content filters. A content filter has two parts:

- a read converter converts from convenience to canonical format
- a write converter converts from canonical to convenience format.

Many of these converters will provide *round-trip* conversion, i.e. applying the read converter followed by the write converter gives back the original content. However, others may provide an asymmetric conversion. For example, a read converter might strip trailing whitespace off lines in source code while the matching write converter might pass content through unchanged.

## **Enabling content filters**

Content filters are typically provided by plugins, so the first step in using them is to install the relevant plugins and read their documentation. Some plugins may be very specific about which files they filter, e.g. only files ending in .java or .php. In other cases, the plugin may leave it in the user's hands to define which files are to be filtered. This is typically done using rule-based preferences. See bzr help rules for general information about defining these.

## Impact on commands

Read converters are only applied to commands that read content from a working tree, e.g. status, diff and commit. For example, bzr diff will apply read converters to files in the working tree, then compare the results to the content last committed.

Write converters are only applied by commands that **create files in a working tree**, e.g. branch, checkout, update. If you wish to see the canonical format of a file or tree, use bzr cat or bzr export respectively.

Note: bzr commit does not implicitly apply write converters after comitting files. If this makes sense for a given plugin providing a content filter, the plugin can usually achieve this effect by using a start\_commit or post\_commit hook say. See *Hooks* for more information on hooks.

## Refreshing your working tree

For performance reasons, Bazaar caches the timestamps of files in a working tree, and assumes files are unchanged if their timestamps match the cached values. As a consequence, there are times when you may need to explicitly ask for content filtering to be reapplied in one or both directions, e.g. after installing or reconfiguring plugins providing it.

Here are some general guidelines for doing this:

- To reapply read converters, touch files, i.e. update their timestamp. Operations like bzr status should then reapply the relevant read converters and compare the end result with the canonical format.
- To reapply write converters, ensure there are no local changes, delete the relevant files and run bzr revert on those files.

Note: In the future, it is likely that additional options will be added to commands to make this refreshing process faster and safer.

# **Criss-Cross**

A criss-cross in the branch history can cause the default merge technique to emit more conflicts than would normally be expected.

In complex merge cases, bzr merge --lca or bzr merge --weave may give better results. You may wish to bzr revert the working tree and merge again. Alternatively, use bzr remerge on particular conflicted files.

Criss-crosses occur in a branch's history if two branches merge the same thing and then merge one another, or if two branches merge one another at the same time. They can be avoided by having each branch only merge from or into a designated central branch (a "star topology").

Criss-crosses cause problems because of the way merge works. Bazaar's default merge is a three-way merger; in order to merge OTHER into THIS, it must find a basis for comparison, BASE. Using BASE, it can determine whether differences between THIS and OTHER are due to one side adding lines, or from another side removing lines.

Criss-crosses mean there is no good choice for a base. Selecting the recent merge points could cause one side's changes to be silently discarded. Selecting older merge points (which Bazaar does) mean that extra conflicts are emitted.

The weave merge type is not affected by this problem because it uses line-origin detection instead of a basis revision to determine the cause of differences.

#### Reference »

# **Diverged Branches**

When Bazaar tries to push one branch onto another, it requires that the destination branch must be ready to receive the source branch. If this isn't the case, then we say that the branches have diverged. Branches are considered diverged if the destination branch's most recent commit is one that has not been merged (directly or indirectly) by the source branch. To recover from diverged branches, one must merge the missing revisions into the source branch.

This situation commonly arises when using a centralized workflow with local commits. If someone else has committed new work to the mainline since your last pull and you have local commits that have not yet been pushed to the mainline, then your local branch and the mainline have diverged.

## Discovering What Has Diverged

The bzr missing command is used to find out what revisions are in another branch that are not present in the current branch, and viceversa. It shows a summary of which extra revisions exist in each branch. If you want to see the precise effects of those revisions, you can use bzr diff --old=other\_branch to show the differences between other branch and your current branch.

### A Solution

The solution is to merge the revisions from the mainline into your local branch. To do so, use bzr merge to get the new revisions from the mainline. This merge may result in conflicts if the other developer's changes overlap with your changes. These conflicts should be resolved before continuing. After any conflicts have been resolved, or even if there were no conflicts, Bazaar requires that you explicitly commit these new revisions to your local branch. This requirement gives you an opportunity to test the resulting working tree for correctness, since the merged revisions could have made arbitrary changes. After testing, you should commit the merge using bzr commit. This clears up the diverged branches situation. Your local branch can now be pushed to the mainline.

Home | Documentation | Table of Contents (2.2b1) » Bazaar User

Reference »

previous | next

## **End of Line Conversion**

EOL conversion is provided as a content filter where Bazaar internally stores a canonical format but outputs a convenience format. See bzr help content-filters for general information about using these.

Note: Content filtering is only supported in recently added formats, e.g. 1.14. Be sure that both the repository *and* the branch are in a recent format. (Just setting the format on the repository is not enough.) If content filtering does not appear to be working, use 'bzr info -v' to confirm that the branch is using "Working tree format 5" or later.

EOL conversion needs to be enabled for selected file patterns using rules. See bzr help rules for general information on defining rules. Currently, rules are only supported in \$BZR\_HOME/.bazaar/rules (or %BZR\_HOME%/bazaar/2.0/rules on Windows). Branch specific rules will be supported in a future verison of Bazaar.

To configure which files to filter, set eol to one of the values below. (If a value is not set, exact is the default.)

Value	Checkout end-of-lines as	Commit end-of- lines as
native	crlf on Windows, lf otherwise	lf
1f	lf	1f
crlf	crlf	lf
exact	No conversion	Exactly as in file

Note: For safety reasons, no conversion is applied to any file where a null byte is detected in the file.

For users working on a cross-platform project, here is a suggested rule to use as a starting point:

```
[name *]
eol = native
```

If you have binary files that do not contain a null byte though, be sure to add eol = exact rules for those as well. You can do this by giving more explicit patterns earlier in the rules file. For example:

```
[name *.png]
eol = exact

[name *]
eol = native
```

If your working tree is on a network drive shared by users on different operating systems, you typically want to force certain conventions for certain files. In that way, if a file is created with the wrong line endings or line endings get mixed during editing, it gets committed correctly and gets checked out correctly. For example:

```
[name *.bat]
eol = crlf

[name *.sh]
eol = lf

[name *]
eol = native
```

If you take the care to create files with their required endings, you can achieve *almost* the same thing by using eol = exact. It is slightly safer to use If and crlf though because edits accidentally introducing mixed line endings will be corrected during commit for files with those settings.

If you have sample test data that deliberately has text files with mixed newline conventions, you can ask for those to be left alone like this:

```
[name test_data/]
eol = exact

[name *]
eol = native
```

Note that exact does not imply the file is binary but it does mean that no conversion of end-of-lines will be done. (Bazaar currently relies of content analysis to detect binary files for commands like diff. In the future, a binary = true rule may be added but it is not supported yet.)

If you have an existing repository with text files already stored using Windows newline conventions (crlf), then you may want to keep using that convention in the repository. Forcing certain files to this convention may also help users who do not have rules configured. To do this, set eol to one of the values below.

Value	Checkout end-of- lines as	Commit end-of- lines as
native-with-crlf-in- repo	crlf on Windows, lf otherwise	crlf
lf-with-crlf-in-repo	lf	crlf
crlf-with-crlf-in- repo	crlf	crlf

For users working on an existing project that uses Windows newline conventions in their Bazaar repository, this rule is suggested as a starting point:

```
[name *]
eol = native-with-crlf-in-repo
```

For new projects, it is recommended that end-of-lines be stored as 1f and that users stick to the basic settings, i.e. native, 1f, crlf and exact.

Note: Bazaar's EOL conversion will convert the content of files but never reject files because a given line ending or mixed line endings are found. A precommit hook should be used if you wish to validate (and not just convert) content before committing.

# Storage Formats

To ensure that older clients do not access data incorrectly, Bazaar's policy is to introduce a new storage format whenever new features requiring new metadata are added. New storage formats may also be introduced to improve performance and scalability.

The newest format, 2a, is highly recommended. If your project is not using 2a, then you should suggest to the project owner to upgrade.

**Note:** Some of the older formats have two variants: a plain one and a rich-root one. The latter include an additional field about the root of the tree. There is no performance cost for using a rich-root format but you cannot easily merge changes from a rich-root format into a plain format. As a consequence, moving a project to a rich-root format takes some co-ordination in that all contributors need to upgrade their repositories around the same time. 2a and all future formats will be implicitly rich-root.

See *Current Storage Formats* for the complete list of currently supported formats. See *Other Storage Formats* for descriptions of any available experimental and deprecated formats.

### **Patterns**

Bazaar uses patterns to match files at various times. For example, the add command skips over files that match ignore patterns and preferences can be associated with files using rule patterns. The pattern syntax is described below.

Trailing slashes on patterns are ignored. If the pattern contains a slash or is a regular expression, it is compared to the whole path from the branch root. Otherwise, it is compared to only the last component of the path. To match a file only in the root directory, prepend './'. Patterns specifying absolute paths are not allowed.

Patterns may include globbing wildcards such as:

```
? - Matches any single character except '/'
* - Matches 0 or more characters except '/'
/**/ - Matches 0 or more directories in a path
[a-z] - Matches a single character from within a group of chara
```

Patterns may also be Python regular expressions. Regular expression patterns are identified by a 'RE:' prefix followed by the regular expression. Regular expression patterns may not include named or numbered groups.

Ignore patterns may be prefixed with '!', which means that a filename matched by that pattern will not be ignored.

# Repositories

Repositories in Bazaar are where committed information is stored. There is a repository associated with every branch.

Repositories are a form of database. Bzr will usually maintain this for good performance automatically, but in some situations (e.g. when doing very many commits in a short time period) you may want to ask bzr to optimise the database indices. This can be done by the 'bzr pack' command.

By default just running 'bzr init' will create a repository within the new branch but it is possible to create a shared repository which allows multiple branches to share their information in the same location. When a new branch is created it will first look to see if there is a containing shared repository it can use.

When two branches of the same project share a repository, there is generally a large space saving. For some operations (e.g. branching within the repository) this translates in to a large time saving.

To create a shared repository use the init-repository command (or the alias init-repo). This command takes the location of the repository to create. This means that 'bzr init-repository repo' will create a directory named 'repo', which contains a shared repository. Any new branches that are created in this directory will then use it for storage.

It is a good idea to create a repository whenever you might create more than one branch of a project. This is true for both working areas where you are doing the development, and any server areas that you use for hosting projects. In the latter case, it is common to want branches without working trees. Since the files in the branch will not be edited directly there is no need to use up disk space for a

working tree. To create a repository in which the branches will not have working trees pass the '-no-trees' option to 'init-repository'.

#### Related commands:





# Rules

### Introduction

Rules are defined in ini file format where the sections are file glob patterns and the contents of each section are the preferences for files matching that pattern(s). For example:

```
[name *.bat]
eol = dos

[name *.html *.xml]
keywords = xml_escape
```

Preferences like these are useful for commands and plugins wishing to provide custom behaviour for selected files. For more information on end of line conversion see *End of Line Conversion*. Keyword support is provided by the keywords plugin.

# Files

Default rules for all branches are defined in the optional file  ${\tt BZR\_HOME/rules}.$ 

## Rule Patterns

Patterns are ordered and searching stops as soon as one matches. As a consequence, more explicit patterns should be placed towards the top of the file. Rule patterns use exactly the same conventions as ignore patterns. See *Patterns* for details.

**Note:** Patterns containing square brackets or spaces should be surrounded in quotes to ensure they are correctly parsed.

Home | Documentation | Table of Contents (2.2b1) » Bazaar User

Reference »

previous | next

# Standalone Trees

A standalone tree is a working tree with an associated repository. It is an independently usable branch, with no dependencies on any other. Creating a standalone tree (via bzr init) is the quickest way to put an existing project under version control.

#### **Related Commands:**

init Make a directory into a versioned branch.

# Branches Out of Sync

When reconfiguring a checkout, tree or branch into a lightweight checkout, a local branch must be destroyed. (For checkouts, this is the local branch that serves primarily as a cache.) If the branch-to-be-destroyed does not have the same last revision as the new reference branch for the lightweight checkout, data could be lost, so Bazaar refuses.

How you deal with this depends on *why* the branches are out of sync.

If you have a checkout and have done local commits, you can get back in sync by running "bzr update" (and possibly "bzr commit").

If you have a branch and the remote branch is out-of-date, you can push the local changes using "bzr push". If the local branch is out of date, you can do "bzr pull". If both branches have had changes, you can merge, commit and then push your changes. If you decide that some of the changes aren't useful, you can "push —overwrite" or "pull —overwrite" instead.

Home | Documentation | Table of Contents (2.2b1) » Bazaar User

Reference »

previous | next

# **Working Trees**

A working tree is the contents of a branch placed on disk so that you can see the files and edit them. The working tree is where you make changes to a branch, and when you commit the current state of the working tree is the snapshot that is recorded in the commit.

When you push a branch to a remote system, a working tree will not be created. If one is already present the files will not be updated. The branch information will be updated and the working tree will be marked as out-of-date. Updating a working tree remotely is difficult, as there may be uncommitted changes or the update may cause content conflicts that are difficult to deal with remotely.

If you have a branch with no working tree you can use the 'checkout' command to create a working tree. If you run 'bzr checkout .' from the branch it will create the working tree. If the branch is updated remotely, you can update the working tree by running 'bzr update' in that directory.

If you have a branch with a working tree that you do not want the 'remove-tree' command will remove the tree if it is safe. This can be done to avoid the warning about the remote working tree not being updated when pushing to the branch. It can also be useful when working with a '-no-trees' repository (see 'bzr help repositories').

If you want to have a working tree on a remote machine that you push to you can either run 'bzr update' in the remote branch after each push, or use some other method to update the tree during the push. There is an 'rspush' plugin that will update the working tree using rsync as well as doing a push. There is also a 'push-and-update' plugin that automates running 'bzr update' via SSH after each push.

#### Useful commands:

checkout remove-tree Removes the working tree when a branch does not have update When a working tree from a branch when it is When a working tree is out of sync with it's assoc this will update the tree to match the branch.

# **Authentication Settings**

#### Intent

Many different authentication policies can be described in the authentication.conf file but a particular user should need only a few definitions to cover his needs without having to specify a user and a password for every branch he uses.

The definitions found in this file are used to find the credentials to use for a given url. The same credentials can generally be used for as many branches as possible by grouping their declaration around the remote servers that need them. It's even possible to declare credentials that will be used by different servers.

The intent is to make this file as small as possible to minimize maintenance.

Once the relevant credentials are declared in this file you may use branch urls without embedding passwords (security hazard) or even users (enabling sharing of your urls with others).

Instead of using:

```
bzr branch ftp://joe:secret@host.com/path/to/my/branch
```

you simply use:

```
bzr branch ftp://host.com/path/to/my/branch
```

provided you have created the following authentication.conf file:

```
[myprojects]
scheme=ftp
host=host.com
user=joe
password=secret
```

### **Authentication definitions**

There are two kinds of authentication used by the various schemes supported by bzr:

#### 1. user and password

representation needs a (user, password) to authenticate against a host sftp can use either a password or a host key to authenticate. However, ssh agents are a better, more secure solution. So we have chosen to not provide our own less secure method.

#### 2. user, realm and password

HTTP and HTTPs needs a (user, realm, password) to authenticate against a host. But, by using .htaccess files, for example, it is possible to define several (user, realm, password) for a given host. So what is really needed is (user, password, host, path). The realm is not taken into account in the definitions, but will displayed if bzr prompts you for a password.

HTTP proxy can be handled as HTTP (or HTTPS) by explicitly specifying the appropriate port.

To take all schemes into account, the password will be deduced from a set of authentication definitions (scheme, host, port, path, user, password).

- scheme: can be empty (meaning the rest of the definition can be used for any scheme), SFTP and bzr+ssh should not be used here, ssh should be used instead since this is the real scheme regarding authentication,
- host: can be empty (to act as a default for any host),

- port can be empty (useful when an host provides several servers for the same scheme), only numerical values are allowed, this should be used only when the server uses a port different than the scheme standard port,
- path: can be empty (FTP or SFTP will never user it),
- user: can be empty (bzr will defaults to python's getpass.get\_user()),
- password: can be empty if you prefer to always be prompted for your password.

Multiple definitions can be provided and, for a given URL, bzr will select a (user [, password]) based on the following rules :

- 1. the first match wins,
- 2. empty fields match everything,
- 3. scheme matches even if decorators are used in the requested URL,
- 4. host matches exactly or act as a domain if it starts with '.' (project.bzr.sf.net will match .bzr.sf.net but projectbzr.sf.net will not match bzr.sf.net).
- 5. port matches if included in the requested URL (exact matches only)
- 6. path matches if included in the requested URL (and by rule #2 above, empty paths will match any provided path).

## File format

The general rules for *configuration files* apply except for the variable policies.

Each section describes an authentication definition.

The section name is an arbitrary string, only the **DEFAULT** value is reserved and should appear as the *last* section.

Each section should define:

user: the login to be used,

Each section could define:

- host: the remote server,
- port: the port the server is listening,
- path: the branch location,
- password: the password.

## **Examples**

## Personal projects hosted outside

All connections are done with the same user (the remote one for which the default bzr one is not appropriate) and the password is always prompted with some exceptions:

```
# Pet projects on hobby.net
[hobby]
host=r.hobby.net
user=jim
password=obvious1234

# Home server
[home]
scheme=https
host=home.net
user=joe
password=lessobV10us

[DEFAULT]
# Our local user is barbaz, on all remote sites we're known as user=foobar
```

### Source hosting provider

In the shp.net (fictitious) domain, each project has its own site:

```
[shpnet domain]
# we use sftp, but ssh is the scheme used for authentication
scheme=ssh
# The leading '.' ensures that 'shp.net' alone doesn't match
host=.shp.net
user=joe
# bzr don't support supplying a password for sftp,
# consider using an ssh agent if you don't want to supply
# a password interactively. (pageant, ssh-agent, etc)
```

## HTTPS, SFTP servers and their proxy

At company.com, the server hosting release and integration branches is behind a proxy, and the two branches use different authentication policies:

```
[reference code]
scheme=https
host=dev.company.com
path=/dev
user=user1
password=pass1
# development branches on dev server
[dev]
scheme=ssh # bzr+ssh and sftp are available here
host=dev.company.com
path=/dev/integration
user=user2
# proxy
[proxy]
scheme=http
host=proxy.company.com
port=3128
user=proxyuser1
password=proxypass1
```

### Planned enhancements

The following are not yet implemented but planned as parts of a work in progress:

- add a password\_encoding field allowing:
  - storing the passwords in various obfuscating encodings (base64 for one),
  - delegate password storage to plugins (.netrc for example).
- update the credentials when the user is prompted for user or password,
- add a verify\_certificates field for HTTPS.

The password\_encoding and verify\_certificates fields are recognized but ignored in the actual implementation.

# Bug Tracker Settings

When making a commit, metadata about bugs fixed by that change can be recorded by using the --fixes option. For each bug marked as fixed, an entry is included in the 'bugs' revision property stating '<url> <status>'. (The only status value currently supported is fixed.)

The --fixes option allows you to specify a bug tracker and a bug identifier rather than a full URL. This looks like:

```
bzr commit --fixes <tracker>:<id>
```

where "<tracker>" is an identifier for the bug tracker, and "<id>" is the identifier for that bug within the bugtracker, usually the bug number.

Bazaar knows about a few bug trackers that have many users. If you use one of these bug trackers then there is no setup required to use this feature, you just need to know the tracker identifier to use. These are the bugtrackers that are built in:

URL	Abbreviation Example	
https://bugs.launchpad.net/	lp	lp:12345
http://bugs.debian.org/	deb	deb:12345
http://bugzilla.gnome.org/	gnome	gnome:12345

For the bug trackers not listed above configuration is required. Support for generating the URLs for any project using Bugzilla or Trac is built in, along with a template mechanism for other bugtrackers with simple URL schemes. If your bug tracker can't be described by one of the schemes described below then you can write a plugin to support it.

If you use Bugzilla or Trac, then you only need to set a configuration

variable which contains the base URL of the bug tracker. These options can go into bazaar.conf, branch.conf or into a branch-specific configuration section in locations.conf. You can set up these values for each of the projects you work on.

Note: As you provide a short name for each tracker, you can specify one or more bugs in one or more trackers at commit time if you wish.

# Launchpad

Use bzr commit --fixes 1p:2 to record that this commit fixes bug 2.

# bugzilla\_<tracker>\_url

If present, the location of the Bugzilla bug tracker referred to by <tracker>. This option can then be used together with bzr commit -fixes to mark bugs in that tracker as being fixed by that commit. For example:

```
bugzilla_squid_url = http://www.squid-cache.org/bugs
```

would allow bzr commit --fixes squid:1234 to mark Squid's bug 1234 as fixed.

# trac\_<tracker>\_url

If present, the location of the Trac instance referred to by <tracker>. This option can then be used together with bzr commit --fixes to mark bugs in that tracker as being fixed by that commit. For example:

```
trac_twisted_url = http://www.twistedmatrix.com/trac
```

would allow bzr commit --fixes twisted:1234 to mark Twisted's bug 1234 as fixed.

# bugtracker\_<tracker>\_url

If present, the location of a generic bug tracker instance referred to by <tracker>. The location must contain an {id} placeholder, which will be replaced by a specific bug ID. This option can then be used together with bzr commit --fixes to mark bugs in that tracker as being fixed by that commit. For example:

```
bugtracker_python_url = http://bugs.python.org/issue{id}
```

would allow bzr commit --fixes python:1234 to mark bug 1234 in Python's Roundup bug tracker as fixed, or:

```
bugtracker_cpan_url = http://rt.cpan.org/Public/Bug/Display.htm
```

for CPAN's RT bug tracker.

# **Configuration Settings**

# **Environment settings**

While most configuration is handled by configuration files, some options which may be semi-permanent can also be controlled through the environment.

#### BZR\_EMAIL

Override the email id used by Bazaar. Typical format:

```
"John Doe <jdoe@example.com>"
```

See also the email configuration value.

## BZR\_PROGRESS\_BAR

Override the progress display. Possible values are "none", "dots", "tty"

## BZR\_SIGQUIT\_PDB

Control whether SIGQUIT behaves normally or invokes a breakin debugger.

- 0 = Standard SIGQUIT behavior (normally, exit with a core dump)
- 1 = Invoke breakin debugger (default)

#### BZR HOME

Override the home directory used by Bazaar.

## BZR\_SSH

Select a different SSH implementation.

## BZR\_PDB

Control whether to launch a debugger on error.

- 0 = Standard behavior
- 1 = Launch debugger

#### BZR REMOTE PATH

Path to the Bazaar executable to use when using the bzr+ssh protocol.

See also the bzr\_remote\_path configuration value.

## BZR\_EDITOR

Path to the editor Bazaar should use for commit messages, etc.

#### BZR PLUGIN PATH

The path to the plugins directory that Bazaar should use. If not set, Bazaar will search for plugins in:

- the user specific plugin directory (containing the user plugins),
- the bzrlib directory (containing the core plugins),
- the site specific plugin directory if applicable (containing the site plugins).

If BZR\_PLUGIN\_PATH is set in any fashion, it will change the the way the plugin are searched.

As for the PATH variables, if multiple directories are specified in

BZR\_PLUGIN\_PATH they should be separated by the platform specific appropriate character (':' on Unix/Linux/etc, ';' on windows)

By default if BZR\_PLUGIN\_PATH is set, it replaces searching in user. However it will continue to search in core and site unless they are explicitly removed.

If you need to change the order or remove one of these directories, you should use special values:

- -user, -core, -site will remove the corresponding path from the default values,
- +user, +core, +site will add the corresponding path before the remaining default values (and also remove it from the default values).

Note that the special values 'user', 'core' and 'site' should be used literally, they will be substituted by the corresponding, platform specific, values.

The examples below use ':' as the separator, windows users should use ';'.

Overriding the default user plugin directory:

```
BZR_PLUGIN_PATH='/path/to/my/other/plugins'
```

Disabling the site directory while retaining the user directory:

```
BZR_PLUGIN_PATH='-site:+user'
```

Disabling all plugins (better achieved with –no-plugins):

```
BZR_PLUGIN_PATH='-user:-core:-site'
```

Overriding the default site plugin directory:

#### BZR DISABLE PLUGINS

Under special circumstances (mostly when trying to diagnose a bug), it's better to disable a plugin (or several) rather than uninstalling them completely. Such plugins can be specified in the BZR DISABLE PLUGINS environment variable.

In that case, bzr will stop loading the specified plugins and will raise an import error if they are explicitly imported (by another plugin that depends on them for example).

Disabling myplugin and yourplugin is achieved by:

BZR\_DISABLE\_PLUGINS='myplugin:yourplugin'

#### BZR PLUGINS AT

When adding a new feature or working on a bug in a plugin, developers often need to use a specific version of a given plugin. Since python requires that the directory containing the code is named like the plugin itself this make it impossible to use arbitrary directory names (using a two-level directory scheme is inconvenient). BZR\_PLUGINS\_AT allows such directories even if they don't appear in BZR\_PLUGIN\_PATH.

Plugins specified in this environment variable takes precedence over the ones in BZR\_PLUGIN\_PATH.

The variable specified a list of plugin\_name@plugin path, plugin\_name being the name of the plugin as it appears in python module paths, plugin\_path being the path to the directory containing the plugin code itself (i.e. plugins/myplugin not plugins). Use ':' as the list

separator, use ';' on windows.

# Example:

Using a specific version of myplugin: BZR\_PLUGINS\_AT='myplugin@/home/me/bugfixes/123456-myplugin

### **BZRPATH**

The path where Bazaar should look for shell plugin external commands.

# Configuration files

#### Location

Configuration files are located in \$HOME/.bazaar on Linux/Unix and C:\Documents and Settings\<username>\Application Data\Bazaar\2.0 on Windows. (You can check the location for your system by using bzr version.)

There are three primary configuration files in this location:

- bazaar.conf describes default configuration options,
- locations.conf describes configuration information for specific branch locations,
- authentication.conf describes credential information for remote servers.

Each branch can also contain a configuration file that sets values specific to that branch. This file is found at <code>.bzr/branch/branch.conf</code> within the branch. This file is visible to all users of a branch, if you wish to override one of the values for a branch with a setting that is specific to you then you can do so in <code>locations.conf</code>.

#### General format

An ini file has three types of contructs: section headers, section variables and comments.

#### Comments

A comment is any line that starts with a "#" (sometimes called a "hash mark", "pound sign" or "number sign"). Comment lines are

ignored by Bazaar when parsing ini files.

#### Section headers

A section header is a word enclosed in brackets that starts at the begining of a line. A typical section header looks like this:

```
[DEFAULT]
```

The only valid section headers for bazaar.conf currently are [DEFAULT] and [ALIASES]. Section headers are case sensitive. The default section provides for setting variables which can be overridden with the branch config file.

For locations.conf, the variables from the section with the longest matching section header are used to the exclusion of other potentially valid section headers. A section header uses the path for the branch as the section header. Some examples include:

```
[http://mybranches.isp.com/~jdoe/branchdir]
[/home/jdoe/branches/]
```

#### Section variables

A section variable resides within a section. A section variable contains a variable name, an equals sign and a value. For example:

#### Variable policies

Variables defined in a section affect the named directory or URL plus any locations they contain. Policies can be used to change how a variable value is interpreted for contained locations. Currently there are three policies available:

#### none:

the value is interpreted the same for contained locations. This is the default behaviour.

#### norecurse:

the value is only used for the exact location specified by the section name.

#### appendpath:

for contained locations, any additional path components are appended to the value.

Policies are specified by keys with names of the form "\$var:policy". For example, to define the push location for a tree of branches, the following could be used:

```
[/top/location]
push_location = sftp://example.com/location
push_location:policy = appendpath
```

With this configuration, the push location for /top/location/branch1 would be sftp://example.com/location/branch1.

## The main configuration file, bazaar.conf

bazaar.conf allows two sections: [DEFAULT] and [ALIASES]. The default section contains the default configuration options for all branches. The default section can be overriden by providing a branch-specific section in locations.conf.

A typical bazaar.conf section often looks like the following:

# The branch location configuration file, locations.conf

locations.conf allows one to specify overriding settings for a specific branch. The format is almost identical to the default section in bazaar.conf with one significant change: The section header, instead of saying default, will be the path to a branch that you wish to override a value for. The '?' and '\*' wildcards are supported:

```
[/home/jdoe/branches/nethack]
email = Nethack Admin <nethack@nethack.com>

[http://hypothetical.site.com/branches/devel-branch]
create_signatures = always
check_signatures = always

[http://bazaar-vcs.org/bzr/*]
check_signatures = require
```

# The authentication configuration file, authentication.conf

authentication.conf allows one to specify credentials for remote servers. This can be used for all the supported transports and any part of bzr that requires authentication (smtp for example).

The syntax of the file obeys the same rules as the others except for the variable policies which don't apply.

For more information on the possible uses of the authentication configuration file see *Authentication Settings*.

# Common variable options

## debug flags

A comma-separated list of debugging options to turn on. The same values can be used as with the -D command-line option (see *help global-options*). For example:

```
debug_flags = hpss
```

#### email

The email address to use when committing a branch. Typically takes the form of:

```
email = Full Name <account@hostname.tld>
```

#### editor

The path of the editor that you wish to use if *bzr commit* is run without a commit message. This setting is trumped by the environment variable BZR\_EDITOR, and overrides the VISUAL and EDITOR environment variables.

## log\_format

The default log format to use. Standard log formats are long, short and line. Additional formats may be provided by plugins. The default value is long.

### check signatures

Defines the behavior for signatures.

#### require

The gnupg signature for revisions must be present and must be valid.

#### ignore

Do not check gnupg signatures of revisions.

#### check-available

(default) If gnupg signatures for revisions are present, check them. Bazaar will fail if it finds a bad signature, but will not fail if no signature is present.

### create\_signatures

Defines the behaviour of signing revisions.

#### always

Sign every new revision that is committed.

#### when-required

(default) Sign newly committed revisions only when the branch requires signed revisions.

#### never

Refuse to sign newly committed revisions, even if the branch requires signatures.

#### recurse

Only useful in locations.conf. Defines whether or not the configuration for this section applies to subdirectories:

#### true

(default) This section applies to subdirectories as well.

false

This section only applies to the branch at this directory and not branches below it.

### gpg\_signing\_command

(Default: "gpg"). Which program should be used to sign and check revisions. For example:

gpg\_signing\_command = /usr/bin/gnpg

### bzr\_remote\_path

(Default: "bzr"). The path to the command that should be used to run the smart server for bzr. This value may only be specified in locations.conf, because:

- it's needed before branch.conf is accessible
- allowing remote branch.conf files to specify commands would be a security risk

It is overridden by the BZR\_REMOTE\_PATH environment variable.

### smtp\_server

(Default: "localhost"). SMTP server to use when Bazaar needs to send email, eg. with merge-directive --mail-to, or the bzr-email plugin.

### smtp\_username, smtp\_password

User and password to authenticate to the SMTP server. If smtp\_username is set, and smtp\_password is not, Bazaar will prompt for a password. These settings are only needed if the SMTP server requires authentication to send mail.

## mail\_client

A mail client to use for sending merge requests. By default, bzr will try to use mapi on Windows. On other platforms, it will try xdg-email. If either of these fails, it will fall back to editor.

Supported values for specific clients:

**claws:** Use Claws. This skips a dialog for attaching files.

evolution: Use Evolution.

kmail: Use KMail.mutt: Use Mutt.

Use Mozilla Thunderbird or Icedove. For

thunderbird: Thunderbird/Icedove 1.5, this works around some

bugs that xdg-email doesn't handle.

Supported generic values are:

default: See above.

Use your editor to compose the merge request. This also

editor: uses your commit id, (see bzr whoami), smtp server and

(optionally) smtp\_username and smtp\_password.

mapi: Use your preferred e-mail client on Windows.

xdq-

email:

Use xdg-email to run your preferred mail program

## submit\_branch

The branch you intend to submit your current work to. This is automatically set by bzr send, and is also used by the submit: revision spec. This should usually be set on a per-branch or per-location basis.

## public\_branch

A publically-accessible version of this branch (implying that this version is not publically-accessible). Used (and set) by bzr send.

## suppress\_warnings

A list of strings, each string represent a warning that can be emitted by bzr. Mentioning a warning in this list tells bzr to not emit it.

#### Valid values:

#### • format\_deprecation:

whether the format deprecation warning is shown on repositories that are using deprecated formats.

## Branch type specific options

These options apply only to branches that use the dirstate-tags or later format. They are usually set in .bzr/branch/branch.conf automatically, but may be manually set in locations.conf or bazaar.conf.

## append\_revisions\_only

If set to "True" then revisions can only be appended to the log, not removed. A branch with this setting enabled can only pull from another branch if the other branch's log is a longer version of its own. This is normally set by bzr init --append-revisions-only.

### parent\_location

If present, the location of the default branch for pull or merge. This option is normally set by pull --remember or merge --remember.

## push\_location

If present, the location of the default branch for push. This option is normally set by push --remember.

### push strict

If present, defines the --strict option default value for checking uncommitted changes before pushing.

### dpush strict

If present, defines the --strict option default value for checking uncommitted changes before pushing into a different VCS without any custom bzr metadata.

## bound\_location

The location that commits should go to when acting as a checkout. This option is normally set by bind.

#### bound

If set to "True", the branch should act as a checkout, and push each commit to the bound\_location. This option is normally set by bind/unbind.

### send\_strict

If present, defines the --strict option default value for checking uncommitted changes before sending a merge directive.

## **Conflict Types**

Some operations, like merge, revert and pull, modify the contents of your working tree. These modifications are programmatically generated, and so they may conflict with the current state of your working tree.

When conflicts are present in your working tree (as shown by bzr conflicts), you should resolve them and then inform bzr that the conflicts have been resolved.

Resolving conflicts is sometimes not obvious. Either because the user that should resolve them is not the one responsible for their occurrence, as is the case when merging other people's work or because some conflicts are presented in a way that is not easy to understand.

Bazaar tries to avoid conflicts; its aim is to ask you to resolve the conflict if and only if there's an actual conceptual conflict in the source tree. Because Bazaar doesn't understand the real meaning of the files being versioned, it can, when faced with ambiguities, fall short in either direction trying to resolve the conflict itself. Many kinds of changes can be combined programmatically, but sometimes only a human can determine the right thing to do.

When Bazaar generates a conflict, it adds information into the working tree to present the conflicting versions, and it's up to you to find the correct resolution.

Whatever the conflict is, resolving it is roughly done in two steps:

```
- modify the working tree content so that the conflicted item i
```

state you want to keep,

• inform Bazaar that the conflict is now solved and ask to cleanup any remaining generated information (bzr resolve <item>).

For most conflict types, there are some obvious ways to modify the working tree and put it into the desired state. For some types of conflicts, Bazaar itself already made a choice, when possible.

Yet, whether Bazaar makes a choice or not, there are some other simple but different ways to resolve the conflict.

Each type of conflict is explained below, and the action which must be done to resolve the conflict is outlined.

Various actions are available depending on the kind of conflict, for some of these actions, Bazaar can provide some help. In the end you should at least inform Bazaar that you're done with the conflict with:

```
``bzr resolve FILE --action=done'
```

Note that this is the default action when a single file is involved so you can simply use:

```
``bzr resolve FILE``
```

See bzr help resolve for more details.

### Text conflicts

#### Typical message:

```
Text conflict in FILE
```

These are produced when a text merge cannot completely reconcile two sets of text changes. Bazaar will emit files for each version with the extensions THIS, OTHER, and BASE. THIS is the version of the file from the target tree, i.e. the tree that you are merging changes into. OTHER is the version that you are merging into the target. BASE is an older version that is used as a basis for comparison.

In the main copy of the file, Bazaar will include all the changes that it could reconcile, and any un-reconciled conflicts are surrounded by "herringbone" markers like <<<<<.

Say the initial text is "The project leader released it.", and THIS modifies it to "Martin Pool released it.", while OTHER modifies it to "The project leader released Bazaar." A conflict would look like this:

```
<<<<< TREE
Martin Pool released it.
======
The project leader released Bazaar.
>>>>> MERGE-SOURCE
```

The correct resolution would be "Martin Pool released Bazaar."

You can handle text conflicts either by editing the main copy of the file, or by invoking external tools on the THIS, OTHER and BASE versions. It's worth mentioning that resolving text conflicts rarely involves picking one set of changes over the other. More often, the two sets of changes must be intelligently combined.

If you edit the main copy, be sure to remove the herringbone markers. When you are done editing, the file should look like it never had a conflict, and be ready to commit.

When you have resolved text conflicts, just run bzr resolve --auto, and Bazaar will auto-detect which conflicts you have resolved.

When the conflict is resolved, Bazaar deletes the previously generated <code>.base</code>, <code>.this</code> and <code>.other</code> files if they are still present in the working tree.

### Content conflicts

#### Typical message:

#### Contents conflict in FILE

This conflict happens when there are conflicting changes in the working tree and the merge source, but the conflicted items are not text files. They may be binary files, or symlinks, or directories. It can even happen with files that are deleted on one side, and modified on the other.

Like text conflicts, Bazaar will emit THIS, OTHER and BASE files. (They may be regular files, symlinks or directories). But it will not include a "main copy" of the file with herringbone conflict markers. It will appear that the "main copy" has been renamed to THIS or OTHER.

To resolve that kind of conflict, you should rebuild FILE from either version or a combination of both.

bzr resolve recognizes the following actions:

- --action=take-this will issue bzr mv FILE.THIS FILE,
- --action=take-other Will issue bzr mv FILE.OTHER FILE,
- --action=done will just mark the conflict as resolved.

Any action will also delete the previously generated .BASE, .THIS and .OTHER files if they are still present in the working tree.

## Tag conflicts

#### Typical message:

```
Conflicting tags: version-0.1
```

When pulling from or pushing to another branch, Bazaar informs you about tags that conflict between the two branches; that is the same tag points to two different revisions. You need not resolve these conflicts, but subsequent uses of pull or push will result in the same message.

To resolve the conflict, you must apply the correct tags to either the target branch or the source branch as appropriate. Use "bzr tags – show-ids -d SOURCE\_URL" to see the tags in the source branch. If you want to make the target branch's tags match the source branch, then in the target branch do bzr tag --force -r revid:REVISION\_ID CONFLICTING\_TAG for each of the CONFLICTING\_TAGs, where REVISION\_ID comes from the list of tags in the source branch. You need not call "bzr resolve" after doing this. To resolve in favor of the target branch, you need to similarly use tag --force in the source branch. (Note that pulling or pushing using —overwrite will overwrite all tags as well.)

## Duplicate paths

#### Typical message:

```
Conflict adding file FILE. Moved existing file to FILE.moved.
```

Sometimes Bazaar will attempt to create a file using a pathname that has already been used. The existing file will be renamed to "FILE.moved".

To resolve that kind of conflict, you should rebuild FILE from either version or a combination of both.

bzr resolve recognizes the following actions:

Note that you must get rid of FILE.moved before using -- action=done.

## Unversioned parent

#### Typical message:



Sometimes Bazaar will attempt to create a file whose parent directory is not versioned. This happens when the directory has been deleted in the target, but has a new child in the source, or vice versa. In this situation, Bazaar will version the parent directory as well. Resolving this issue depends very much on the particular scenario. You may wish to rename or delete either the file or the directory. When you are satisfied, you can run "bzr resolve FILE" to mark the conflict as resolved.

## Missing parent

#### Typical message:

```
Conflict adding files to FILE. Created directory.
```

This happens when a directory has been deleted in the target, but has new children in the source. This is similar to the "unversioned parent" conflict, except that the parent directory does not *exist*, instead of just being unversioned. In this situation, Bazaar will create the missing parent. Resolving this issue depends very much on the particular scenario.

To resolve that kind of conflict, you should either remove or rename the children or the directory or a combination of both.

bzr resolve recognizes the following actions:

```
- ``--action=take-this`` will issue ``bzr rm directory`` includ - ``--action=take-other`` will acknowledge Bazaar choice to kee
```

and restoring the directory,

• --action=done will just mark the conflict as resolved.

## **Deleting parent**

#### Typical message:

```
Conflict: can't delete DIR because it is not empty. Not deleti
```

This is the opposite of "missing parent". A directory is deleted in the source, but has new children in the target. Bazaar will retain the directory. Resolving this issue depends very much on the particular scenario.

To resolve that kind of conflict, you should either remove or rename the children or the directory or a combination of both.

bzr resolve recognizes the following actions:

```
- ``--action=take-this`` will acknowledge Bazaar choice to keep
- ``--action=take-other`` will issue ``bzr rm directory`` inclu
```

children,

• --action=done will just mark the conflict as resolved.

## Path conflict

#### Typical message:

```
Path conflict: PATH1 / PATH2
```

This happens when the source and target have each modified the name or parent directory of a file. Bazaar will use the path elements from the source.

To resolve that kind of conflict, you just have to decide what name should be retained for the file involved.

bzr resolve recognizes the following actions:

```
- ``--action=take-this`` will revert Bazaar choice and keep ``P.

•]
```

issuing bzr mv PATH2 PATH1,

- --action=take-other will acknowledge Bazaar choice of keeping PATH2,
- --action=done will just mark the conflict as resolved.

## Parent loop

#### Typical message:

```
Conflict moving FILE into DIRECTORY. Cancelled move.
```

This happens when the source and the target have each moved directories, so that, if the change could be applied, a directory would be contained by itself. For example:

```
$ bzr init
$ bzr mkdir white
$ bzr mkdir black
$ bzr commit -m "BASE"
$ bzr branch . ../other
$ bzr mv white black
$ bzr commit -m "THIS"
$ bzr mv ../other/black ../other/white
$ bzr commit ../other -m "OTHER"
$ bzr merge ../other
```

In this situation, Bazaar will cancel the move, and leave white in black. To resolve that kind of conflict, you just have to decide what name should be retained for the directories involved.

bzr resolve recognizes the following actions:

```
- ``--action=take-this`` will acknowledge Bazaar choice of leav
```

in black,

- --action=take-other will revert Bazaar choice and move black in white by issuing bzr mv black/white white; bzr mv black white,
- --action=done will just mark the conflict as resolved.

## Non-directory parent

#### Typical message:

```
Conflict: foo.new is not a directory, but has files in it.
Created directory.
```

This happens when one side has added files to a directory, and the other side has changed the directory into a file or symlink. For example:

```
$ bzr init
$ bzr mkdir foo
$ bzr commit -m "BASE"
$ bzr branch . ../other
$ rmdir foo
$ touch foo
$ bzr commit -m "THIS"
$ bzr mkdir ../other/foo/bar
$ bzr commit ../other -m "OTHER"
$ bzr merge ../other
```

To resolve that kind of conflict, you have to decide what name should be retained for the file, directory or symlink involved.

bzr resolve recognizes the following actions:

```
- ``--action=take-this`` will issue ``bzr rm --force foo.new``
```

bzr add foo,

- --action=take-other Will issue bzr rm --force foo and bzr mv foo.new foo,
- --action=done will just mark the conflict as resolved.

resolved.

## MalformedTransform

It is possible (though very rare) for Bazaar to raise a MalformedTransform exception. This means that Bazaar encountered a filesystem conflict that it was unable to resolve. This usually indicates a bug. Please let us know if you encounter this. Our bug tracker is at https://launchpad.net/bzr/+bugs

## **Current Storage Formats**

(native) (default) First format for bzr 2.0 series. Uses group-

**2a:** compress storage. Provides rich roots which are a one-way transition.

(native) New in 0.92: Pack-based format with data

**pack-** compatible with dirstate-tags format repositories.

**0.92:** Interoperates with bzr repositories before 0.92 but cannot be read by bzr < 0.92.

**1.14:** (native) A working-tree format that supports content filtering.

1.14-

rich- (native) A variant of 1.14 that supports rich-root data (needed

root: for bzr-svn and bzr-git).

See Storage Formats for more about storage formats.

Home | Documentation | Table of Contents (2.2b1) » Bazaar User

previous | next

Reference »

# Debug Flags

These flags can be passed on the bzr command line or (without the -D prefix) put in the debug\_flags variable in bazaar.conf.

traceback on error.	red				
-Derror Instead of normal error handling, alway traceback on error.  Capture call sites that do expensive or					
traceback on error.  Capture call sites that do expensive or					
-Devii	Instead of normal error handling, always print a traceback on error.				
<u> </u>	Capture call sites that do expensive or badly-scaling operations.				
-Dfetch Trace history copying between reposito	Trace history copying between repositories.				
-Dfilters Emit information for debugging content	filtering.				
-Dforceinvdeltas					
Force use of inventory deltas during ge streaming fetch.	neric				
-Dgraph Trace graph traversal.					
-Dhashcache Log every time a working file is read to hash.	Log every time a working file is read to determine its hash.				
-Dhooks Trace hook execution.	Trace hook execution.				
-Dhpss Trace smart protocol requests and resp	onses.				
-Dhpssdetail More hpss details.	More hpss details.				
-Dhpssvfs Traceback on vfs access to Remote ob	Traceback on vfs access to Remote objects.				
-Dhttp Trace http connections, requests and re	Trace http connections, requests and responses.				
-Dindex Trace major index operations.	Trace major index operations.				
-Dknit Trace knit operations.	Trace knit operations.				
-Dlock Trace when lockdir locks are taken or re	Trace when lockdir locks are taken or released.				
-Dprogress Trace progress bar operations.	Trace progress bar operations.				
-Dmerge Emit information for debugging merges	Emit information for debugging merges.				
-Dno_apport Don't use apport to report crashes.					
-Dno_activity Don't show transport activity indicator in bar.	n progress				
-Dpack Emit information about pack operations	Emit information about pack operations.				

-Drelock	Emit a message every time a branch or repository object is unlocked then relocked the same way.
-Dsftp	Trace SFTP internals.
- Dstatic_tuple	Error when a tuple is used where a StaticTuple is expected
-Dstream	Trace fetch streams.
_ Dstrict_locks	Trace when OS locks are potentially used in a non-portable manner.
-Dunlock	Some errors during unlock are treated as warnings.
-DIDS_never	Never use InterDifferingSerializer when fetching.
-DIDS_always	Always use InterDifferingSerializer to fetch if appropriate for the format, even for non-local fetches.

# **Environment Variables**

BZRPATH	Path where bzr is to look for shell plugin external commands.
BZR_EMAIL	E-Mail address of the user. Overrides EMAIL.
EMAIL	E-Mail address of the user.
BZR_EDITOR	Editor for editing commit messages.  Overrides EDITOR.
EDITOR	Editor for editing commit messages.
BZR_PLUGIN_PATH	Paths where bzr should look for plugins.
BZR_HOME	Directory holding .bazaar config dir. Overrides HOME.
BZR_HOME (Win32)	Directory holding bazaar config dir. Overrides APPDATA and HOME.
BZR_REMOTE_PATH	Full name of remote 'bzr' command (for bzr+ssh:// URLs).
BZR_SSH	Path to SSH client, or one of paramiko, openssh, sshcorp, plink.
BZR_LOG	Location of .bzr.log (use '/dev/null' to suppress log).
BZR_LOG (Win32)	Location of .bzr.log (use 'NUL' to suppress log).
BZR_COLUMNS	Override implicit terminal width.
BZR_CONCURRENCY	Number of processes that can be run concurrently (selftest).

## Files

On Linux: ~/.bazaar/bazaar.conf

On C:\Documents and Settings\username\Application

Windows: Data\bazaar\2.0\bazaar.conf

Contains the user's default configuration. The section [DEFAULT] is used to define general configuration that will be applied everywhere. The section [ALIASES] can be used to create command aliases for commonly used options.

A typical config file might look something like:

```
[DEFAULT]
email=John Doe <jdoe@isp.com>

[ALIASES]
commit = commit --strict
log10 = log --short -r -10..-1
```

Home | Documentation | Table of Contents (2.2b1) » Bazaar User Reference »

previous | next

## **Global Options**

These options may be used with any command, and may appear in front of any command. (e.g. bzr --profile help).

version	Print the version number. Must be supplied before the command.
no- aliases	Do not process command aliases when running this command.
builtin	Use the built-in version of a command, not the plugin version. This does not suppress other plugin effects.
no- plugins	Do not process any plugins.
 concurrency	Number of processes that can be run concurrently (selftest).
profile	Profile execution using the hotshot profiler.
lsprof	Profile execution using the Isprof profiler.
lsprof- file	Profile execution using the Isprof profiler, and write the results to a specified file. If the filename ends with ".txt", text format will be used. If the filename either starts with "callgrind.out" or end with ".callgrind", the output will be formatted for use with KCacheGrind. Otherwise, the output will be a pickle.
coverage	Generate line coverage report in the specified directory.

See <a href="http://doc.bazaar.canonical.com/developers/profiling.html">http://doc.bazaar.canonical.com/developers/profiling.html</a> for more information on profiling.

A number of debug flags are also available to assist troubleshooting and development. See *Debug Flags*.

# Hooks

## Introduction

A hook of type xxx of class yyy needs to be registered using:

```
yyy.hooks.install_named_hook("xxx", ...)
```

See Using hooks in the User Guide for examples.

The class that contains each hook is given before the hooks it supplies. For instance, BranchHooks as the class is the hooks class for *bzrlib.branch.Branch.hooks*.

Each description also indicates whether the hook runs on the client (the machine where bzr was invoked) or the server (the machine addressed by the branch URL). These may be, but are not necessarily, the same machine.

Plugins (including hooks) are run on the server if all of these is true:

- The connection is via a smart server (accessed with a URL starting with "bzr://", "bzr+ssh://" or "bzr+http://", or accessed via a "http://" URL when a smart server is available via HTTP).
- The hook is either server specific or part of general infrastructure rather than client specific code (such as commit).

## **BranchHooks**

### automatic\_tag\_name

Introduced in: 2.2

Called to determine an automatic tag name for a revision.automatic\_tag\_name is called with (branch, revision\_id) and should return a tag name or None if no tag name could be determined. The first non-None tag name returned will be used.

#### open

Introduced in: 1.8

Called with the Branch object that has been opened after a branch is opened.

## post\_change\_branch\_tip

Introduced in: 1.4

Called in bzr client and server after a change to the tip of a branch is made. post\_change\_branch\_tip is called with a bzrlib.branch.ChangeBranchTipParams. Note that push, pull, commit, uncommit will all trigger this hook.

## post\_commit

Introduced in: 0.15

Called in the bzr client after a commit has completed. post\_commit is called with (local, master, old revno, old revid, new revno,

new\_revid). old\_revid is NULL\_REVISION for the first commit to a branch.

## post\_pull

Introduced in: 0.15

Called after a pull operation completes. post\_pull is called with a bzrlib.branch.PullResult object and only runs in the bzr client.

### post\_push

Introduced in: 0.15

Called after a push operation completes. post\_push is called with a bzrlib.branch.BranchPushResult object and only runs in the bzr client.

### post\_uncommit

Introduced in: 0.15

Called in the bzr client after an uncommit completes. post\_uncommit is called with (local, master, old\_revno, old\_revid, new\_revno, new\_revid) where local is the local branch or None, master is the target branch, and an empty branch receives new\_revno of 0, new\_revid of None.

## pre\_change\_branch\_tip

Introduced in: 1.6

Called in bzr client and server before a change to the tip of a branch is made. pre\_change\_branch\_tip is called with a bzrlib.branch.ChangeBranchTipParams. Note that push, pull,

commit, uncommit will all trigger this hook.

### pre\_commit

Introduced in: 0.91

Called after a commit is calculated but before it is is completed. pre\_commit is called with (local, master, old\_revno, old\_revid, future\_revno, future\_revid, tree\_delta, future\_tree). old\_revid is NULL\_REVISION for the first commit to a branch, tree\_delta is a TreeDelta object describing changes from the basis revision. hooks MUST NOT modify this delta. future\_tree is an in-memory tree obtained from CommitBuilder.revision\_tree() and hooks MUST NOT modify this tree.

### set\_rh

Introduced in: 0.15

Invoked whenever the revision history has been set set revision history. The api signature is (branch, revision history), and the branch will be write-locked. The set rh hook can be expensive for bzr to trigger, a better hook to use is Branch.post change branch tip.

## transform\_fallback\_location

Introduced in: 1.9

Called when a stacked branch is activating its fallback locations. transform\_fallback\_location is called with (branch, url), and should return a new url. Returning the same url allows it to be used as-is, returning a different one can be used to cause the branch to stack on a closer copy of that fallback\_location. Note that the branch cannot have history accessing methods called on it during this hook

because the fallback locations have not been activated. When there are multiple hooks installed for transform\_fallback\_location, all are called with the url returned from the previous hook. The order is however undefined.

## BzrDirHooks

## pre\_open

Introduced in: 1.14

Invoked before attempting to open a BzrDir with the transport that the open will use.

## CommandHooks

## extend command

Introduced in: 1.13

Called after creating a command object to allow modifications such as adding or removing options, docs etc. Called with the new bzrlib.commands.Command object.

### get command

Introduced in: 1.17

Called when creating a single command. Called with (cmd\_or\_None, command\_name). get\_command should either return the cmd\_or\_None parameter, or a replacement Command object that should be used for the command. Note that the Command.hooks hooks are core infrastructure. Many users will prefer to use bzrlib.commands.register\_command or plugin\_cmds.register\_lazy.

## get missing command

Introduced in: 1.17

Called when creating a single command if no command could be found. Called with (command\_name). get\_missing\_command should either return None, or a Command object to be used for the command.

### list commands

Introduced in: 1.17

Called when enumerating commands. Called with a set of cmd\_name strings for all the commands found so far. This set is safe to mutate - e.g. to remove a command. list\_commands should return the updated set of command names.

## InfoHooks

## repository

Introduced in: 1.15

Invoked when displaying the statistics for a repository. repository is called with a statistics dictionary as returned by the repository and a file-like object to write to.

## LockHooks

## lock\_acquired

Introduced in: 1.8

Called with a bzrlib.lock.LockResult when a physical lock is acquired.

## lock\_broken

Introduced in: 1.15

Called with a bzrlib.lock.LockResult when a physical lock is broken.

## lock\_released

Introduced in: 1.8

Called with a bzrlib.lock.LockResult when a physical lock is released.

## MergeHooks

## merge\_file\_content

Introduced in: 2.1

Called with a bzrlib.merge.Merger object to create a per file merge object when starting a merge. Should return either None or a subclass of bzrlib.merge.AbstractPerFileMerger. Such objects will then be called per file that needs to be merged (including when one side has deleted the file and the other has changed it). See the AbstractPerFileMerger API docs for details on how it is used by merge.

# MergeDirectiveHooks

## merge\_request\_body

Introduced in: 1.15.0

Called with a MergeRequestBodyParams when a body is needed for a merge request. Callbacks must return a body. If more than one callback is registered, the output of one callback is provided to the next.

## MessageEditorHooks

## commit message template

Introduced in: 1.10

Called message being generated. when a commit is commit message template is called with the bzrlib.commit.Commit object and the message is far. that known S0 commit message template must return a new message to use (which could be the same as it was given. When there are multiple hooks registered for commit message template, they are chained with the result from the first passed into the second, and so on.

## MutableTreeHooks

## post\_commit

Introduced in: 2.0

Called after a commit is performed on a tree. The hook is called with a bzrlib.mutabletree.PostCommitHookParams object. The mutable tree the commit was performed on is available via the mutable\_tree attribute of that object.

## start\_commit

Introduced in: 1.4

Called before a commit is performed on a tree. The start commit hook is able to change the tree before the commit takes place. start\_commit is called with the bzrlib.mutabletree.MutableTree that the commit is being performed on.

## **SmartClientHooks**

### call

Introduced in: unknown

Called when the smart client is submitting a request to the smart server. Called with a bzrlib.smart.client.CallHookParams object. Streaming request bodies, and responses, are not accessible.

## **SmartServerHooks**

### server\_started

Introduced in: 0.16

Called by the bzr server when it starts serving a directory. server\_started is called with (backing urls, public url), where backing\_url is a list of URLs giving the server-specific directory locations, and public\_url is the public URL for the directory being served.

## server\_started\_ex

Introduced in: 1.17

Called by the bzr server when it starts serving a directory. server\_started is called with (backing\_urls, server\_obj).

### server\_stopped

Introduced in: 0.16

Called by the bzr server when it stops serving a directory. server\_stopped is called with the same parameters as the server started hook: (backing urls, public url).

## RioVersionInfoBuilderHooks

## revision

Introduced in: 1.15

Invoked when adding information about a revision to the RIO stanza that is printed. revision is called with a revision object and a RIO stanza.

Home | Documentation | Table of Contents (2.2b1) » Bazaar User
Reference »

previous | next

## Location aliases

Bazaar defines several aliases for locations associated with a branch. These can be used with most commands that expect a location, such as *bzr push*.

#### The aliases are:

```
:parent
:submit the submit branch for this branch
:public the public location of this branch
:bound the branch this branch is bound to, for bound branch
:push the saved location used for `bzr push` with no argum
:this branch
```

For example, to push to the parent location:

```
bzr push :parent
```

# Log Formats

A log format controls how information about each revision is displayed. The standard log formats are compared below:

Feature	long	short	line
design goal committer author date-time format commit message tags merges indicator status/delta diff/patch revision-id branch nick foreign vcs properties	detailed view name+email name+email full full yes - optional optional optional yes yes	concise view name only date only full yes yes optional optional optional - yes	1 revision name only date only top line yes
preferred levels	all	1	1

The default format is long. To change this, define the log\_format setting in the [DEFAULT] section of bazaar.conf like this (say):

```
[DEFAULT]
log_format = short
```

Alternatively, to change the log format used for a given query, use the –long, –short or –line options.

If one of the standard log formats does not meet your needs, additional formats can be provided by plugins.

# Other Storage Formats

No experimental formats are available.

No deprecated formats are available.

See Storage Formats for more about storage formats.

Home | Documentation | Table of Contents (2.2b1) » Bazaar User Reference »

previous | next

## **Revision Identifiers**

A revision identifier refers to a specific state of a branch's history. It can be expressed in several ways. It can begin with a keyword to unambiguously specify a given lookup type; some examples are 'last:1', 'before:yesterday' and 'submit:'.

Alternately, it can be given without a keyword, in which case it will be checked as a revision number, a tag, a revision id, a date specification, or a branch specification, in that order. For example, 'date:today' could be written as simply 'today', though if you have a tag called 'today' that will be found first.

If 'REV1' and 'REV2' are revision identifiers, then 'REV1..REV2' denotes a revision range. Examples: '3647..3649', 'date:yesterday..-1' and 'branch:/path/to/branch1/..branch:/branch2' (note that there are no quotes or spaces around the '..').

Ranges are interpreted differently by different commands. To the "log" command, a range is a sequence of log messages, but to the "diff" command, the range denotes a change between revisions (and not a sequence of changes). In addition, "log" considers a closed range whereas "diff" and "merge" consider it to be open-ended, that is, they include one end but not the other. For example: "bzr log -r 3647..3649" shows the messages of revisions 3647, 3648 and 3649, while "bzr diff -r 3647..3649" includes the changes done in revisions 3648 and 3649, but not 3647.

The keywords used as revision selection methods are the following:

**revid:** Selects a revision using the revision id.

submit: Selects a common ancestor with the submit branch.
 ancestor: Selects a common ancestor with a second branch.
 date: Selects a revision on the basis of a datestamp.

**branch:** Selects the last revision of a specified branch.

**tag:** Selects a revision identified by a tag name.

**revno:** Selects a revision using a number.

**before:** Selects the parent of the revision specified.

**last:** Selects the nth revision from the end.

In addition, plugins can provide other keywords.

A detailed description of each keyword is given below.

Supply a specific revision id, that can be used to specify any ancestry of the branch. Including merges, and pending merg

#### revid:

```
revid:aaaa@bbbb-123456789 -> Select revision 'aaaa@bbbb
```

Diffing against this shows all the changes that were made in is a good predictor of what merge will do. The submit branch bundle and merge directive commands. If no submit branch parent branch is used instead.

#### submit:

The common ancestor is the last revision that existed in Usually this is the branch point, but it could also be a merged.

Examples:

```
$ bzr diff -r submit:
```

Supply the path to a branch to select the common ancestor. The common ancestor is the last revision that existed in Usually this is the branch point, but it could also be a merged.

This is frequently used with 'diff' to return all of the changes ancestor: introduces, while excluding the changes that you have not remote branch.

Examples:

```
ancestor:/path/to/branch
$ bzr diff -r ancestor:../../mainline/branch
```

Supply a datestamp to select the first revision that matche can be 'yesterday', 'today', 'tomorrow' or a YYYY-MM-DD str first entry after a given date (either at midnight or at a specifi One way to display all the changes since yesterday would be

#### date:

```
bzr log -r date:yesterday..
```

#### **Examples:**

```
date:yesterday -> select the first revision date:2006-08-14,17:10:14 -> select the first revision August 14th, 2006 at 5:10p
```

Supply the path to a branch to select its last revision. Examples:

#### branch:

```
branch:/path/to/branch
```

#### tag:

Tags are stored in the branch and created by the 'tag' commuse an integer to specify a revision in the history of the branch can be specified. A negative number will count fror branch (-1 is the last revision, -2 the previous one). If the ne larger than the branch's history, the first revision is returned.

#### revno:

```
revno:1 -> return the first revision
revno:3:/path/to/branch -> return the 3rd revision of
the branch '/path/to/branc
revno:-1 -> The last revision in a bra
-2:http://other/branch -> The second to last revisio
remote branch.
-1000000 -> Most likely the first revi
your history is very long.
```

Supply any revision spec to return the parent of that revision useful when inspecting revisions that are not in the revisionsch.

It is an error to request the parent of the null revision (before Examples:

#### before:

Supply a positive number to get the nth revision from the same as supplying negative numbers to the 'revno:' spec. Ex

#### last:

```
last:1     -> return the last revision
last:3     -> return the revision 2 before the end.
```

# **Standard Options**

Standard options are legal for all commands.

```
--help, -h Show help message.

--verbose, -v Display more information.

--quiet, -q Only display errors and warnings.
```

Unlike global options, standard options can be used in aliases.

# Status Flags

Status flags are used to summarise changes to the working tree in a concise manner. They are in the form:

```
xxx <filename>
```

where the columns' meanings are as follows.

#### Column 1 - versioning/renames:

```
+ File versioned
- File unversioned
R File renamed
? File unknown
X File nonexistent (and unknown to bzr)
C File has conflicts
P Entry for a pending merge (not a file)
```

#### Column 2 - contents:

```
N File created
D File deleted
K File kind changed
M File modified
```

#### Column 3 - execute:

```
* The execute bit was changed
```

## **URL** Identifiers

#### Supported URL prefixes:

```
Access using active FTP.
aftp://
bzr://
                    Fast access using the Bazaar smart server.
                    Fast access using the Bazaar smart server o
bzr+ssh://
file://
                    Access using the standard filesystem (defau
ftp://
                    Access using passive FTP.
                    Read-only access of branches exported on th
http://
https://
                    Read-only access of branches exported on th
sftp://
                    Access using SFTP (most SSH servers provide
4
```

Bazaar supports all of the standard parts within the URL:

```
col>://[user[:password]@]host[:port]/[path]
```

allowing URLs such as:

```
http://bzruser:BadPass@bzr.example.com:8080/bzr/trunk
```

For bzr+ssh:// and sftp:// URLs, Bazaar also supports paths that begin with '~' as meaning that the rest of the path should be interpreted relative to the remote user's home directory. For example if the user remote has a home directory of /home/remote on the server shell.example.com, then:

```
bzr+ssh://remote@shell.example.com/~/myproject/trunk
```

would refer to /home/remote/myproject/trunk.

Many commands that accept URLs also accept location aliases too. See :Location aliases.

#### Reference »

Home | Documentation | Table of Contents (2.2b1) » Bazaar System previous | next

Administrator's Guide »

# Introduction

Welcome to the Bazaar Version Control System's guide for system administrators. Bazaar is a flexible system that provides many possible options for serving projects in ways that will hopefully meet your needs. If you have requirements that are not met by the current state of the Bazaar ecosystem, please let us know at bazaar@lists.canonical.com or on Launchpad at https://launchpad.net/bzr.

# Scope of this guide

In this guide, we will discuss various techniques for making Bazaar projects available, migrating from other Version Control Systems, browsing code over the Web and combining Bazaar with other tools. In many of these categories, multiple options exist and we will try to explains the costs and benefits of the various options.

The intended audience for this guide is the individuals who administer the computers that will do the serving. Much of the configuration that we will discuss requires administrator privileges and we will not necessarily indicate every point that those privileges are needed. That said, reading this guide can also be very helpful for those who are interested in communicating to the system administrators about the requirements for making full use of Bazaar.

# What you need to run a Bazaar server

Where possible, we will discuss both Unix (including Linux) and Windows server environments. For the purposes of this document, we will consider Mac OS X as a type of Unix.

In general, Bazaar requires only Python 2.4 or greater and the cElementTree package (included in Python 2.5 and later) to run. If you would *optionally* like to be able to access branches using SFTP, you need paramiko and pycrypto.

For maximum performance, Bazaar can make use of compiled versions of some critical components of the code. Pure Python alternatives exist for all of these components, but they may be considerably slower. To compile these extensions, you need a C compiler and the relevant header files from the Python package. On Linux, these may be in a separate package. Other operating systems should have the required headers installed by default.

If you are installing a development version of Bazaar, rather than a released version, you will need Pyrex to create the C extensions. The release tarballs already have the Pyrex-created C files.

Home | Documentation | Table of Contents (2.2b1) » Bazaar System previous | next

Administrator's Guide »

# Simple Setups

Consider the following simple scenario where we will be serving Bazaar branches that live on a single server. Those branches are in the subdirectories of /srv/bzr (or c:\\bzr) and they will all be related to a single project called "ProjectX". ProjectX will have a trunk branch and at least one feature branch. As we get further, we will consider other scenarios, but this will be a sufficiently motivating example.

#### **Smart server**

The simplest possible setup for providing outside access to the branches on the server uses Bazaar's built-in smart server tunneled over SSH so that people who can access your server using SSH can have read and write access to branches on the server. This setup uses the authentication mechanisms of SSH including private keys, and the access control mechanisms of the server's operating system. In particular, using groups on the server, it is possible to provide different access privileges to different groups of developers.

#### Setup

There is no setup required for this on the server, apart from having Bazaar installed and SSH access available to your developers. Using SSH configuration options it is possible to restrict developers from using anything *but* Bazaar on the server via SSH, and to limit what part of the file system they can access.

#### Client

Clients can access the branches using URLs with the bzr+ssh:// prefix. For example, to get a local copy of the ProjectX trunk, a developer could do:

```
$ bzr branch bzr+ssh://server.example.com/srv/bzr/projectx/trun
▶
```

If the developers have write access to the /srv/bzr/projectx directory, then they can create new branches themselves using:

```
$ bzr branch bzr+ssh://server.example.com/srv/bzr/projectx/trun
bzr+ssh://server.example.com/srv/bzr/projectx/feature-gui
```

Of course, if this isn't desired, then developers should not have write access to the /srv/bzr/projectx directory.

## **Further Configuration**

For a project with multiple branches that are all related, it is best to use a shared repository to hold all of the branches. To set this up, do:

```
$ cd /srv/bzr
$ bzr init-repo --no-trees projectx
```

The --no-trees option saves space by not creating a copy of the working files on the server's filesystem. Then, any branch created under /srv/bzr/projectx (see Migration for some ways to do this) will share storage space, which is particularly helpful for branches that have many revisions in common, such as a project trunk and its feature branches.

If Bazaar is not installed on the user's path or not specified in the SSH configuration, then a path can be specified from the client with the BZR\_REMOTE\_PATH environment variable. For example, if the Bazaar executable is installed in /usr/local/bzr-2.0/bin/bzr, then a developer could use:

```
$ BZR_REMOTE_PATH=/usr/local/bzr-2.0/bin/bzr bzr info \
bzr+ssh://server.example.com/srv/bzr/proectx/trunk
```

to get information about the trunk branch. The remote path can also be specified in Bazaar's configuration files for a particular location. See bzr help configuration for more details.

If developers have home directories on the server, they can use /~/ in URLs to refer to their home directory. They can also use /~username/ to refer to the home directory of user username. For

example, if there are two developers alice and bob, then Bob could use:

```
$ bzr log bzr+ssh://server.example.com/~/fix-1023
```

to refer to one of his bug fix branches and:

```
$ bzr log bzr+ssh://server.example.com/~alice/fix-2047
```

to refer to one of Alice's branches. [1]

[1] The version of Bazaar installed on the server must be at least 2.1.0b1 or newer to support /~/ in bzr+ssh URLs.

Home | Documentation | Table of Contents (2.2b1) » Bazaar System previous | next Administrator's Guide »

Home | Documentation | Table of Contents (2.2b1) » Bazaar System previous | next

Administrator's Guide »

# Other Setups

## **Dumb servers**

Bazaar can also serve branches over protocols that know nothing about Bazaar's specific needs. These are called "dumb servers" to distinguish them from Bazaar's native protocol. Currently HTTP, FTP, SFTP and HTTP+WebDAV can be used to read branches remotely. FTP, SFTP and HTTP+WebDAV can be used for writing as well. To use any of these protocols, it is just necessary to provide access to the server's filesystem under /srv/bzr.

For example, for Apache to provide read-only access to the branches in /srv/bzr the configuration may look like this:

```
Alias /code /srv/bzr

<Directory /srv/bzr>

   Options Indexes

# ...

</Directory>
```

and users could use the URL http://server.example.com/code/projectx/trunk to refer to the trunk branch.

Note that SFTP access is often available whenever there is SSH access and it may be a good choice when Bazaar cannot be installed on the server to allow <a href="https://access.com/bzr+ssh://">bzr+ssh://access.com/bzr+s

# Smart server over HTTP(S)

Bazaar can use its native protocol with HTTP requests. Since HTTP is a network protocol that is available on many networks, this can be a good option where SSH access is not possible. Another benefit of this setup is that all of the authentication and access control methods available to the HTTP server (basic, LDAP, ActiveDirectory, etc.) are then available to control access to Bazaar branches. More information about setting up this type of access using Apache and FastCGI or mod\_python or WSGI is in the smart server section of the User's Guide.

#### **Direct Smart Server Access**

The built-in server that is used by bzr+ssh:// access can also be used as a persistent server on a dedicated port. Bazaar's official port is 4155, although the port used can be configured. Further information on running the Bazaar smart server from inetd, or directly from the shell is in the User's Guide. The dedicated Bazaar server does not currently perform any authentication, so this server by default provides read-only access. It can be run with the --allow-writes option, but the smart server does not do any additional access control so this may allow undesired people to make changes to branches. (Which of course can be reverted.) If the user that runs the server has write access to the branches on the filesystem, then anyone with access to port 4155 on the server can make changes to the branches stored there.

Home | Documentation | Table of Contents (2.2b1) » Bazaar System

Administrator's Guide »

previous | next

Home | Documentation | Table of Contents (2.2b1) » Bazaar System previous | next

Administrator's Guide »

# Migration

Migrating between version control systems can be a complicated process, and Bazaar has extensive documentation on the process at <a href="http://doc.bazaar-vcs.org/migration/en">http://doc.bazaar-vcs.org/migration/en</a> and we won't attempt to repeat that here. We will try to give a few motivating examples for conversion from Mercurial and Subversion.

## **Fast Import**

In many projects wishing to use Bazaar, there is pre-existing history for the codebase that should be taken into consideration. Bazaar leverages an interchange format originally developed for Git called fast-import to provide migration strategies for many other version control systems. To work with fast-import files, Bazaar needs the fastimport plugin. This can be installed as with any Bazaar plugin.

The way that fast-import can be used for migration is to export the existing history into a fast-import file, then use the bzr fast-import command. The fastimport plugin includes exporters for Subversion, CVS, Git, Mercurial and darcs, accessible as the fast-export-from-xxx commands. Note that fast-import should not be used in a branch with existing history.

Assuming that ProjectX was first developed in Mercurial before switching to Bazaar, and that the Mercurial repository is in /srv/hg/projectx, the following commands will import that history into a newly created trunk branch. (Recall that in Further Configuration we created the /srv/bzr/projectx directory as a shared repository.)

```
$ cd /srv/bzr/projectx
$ bzr fast-export-from-hg ../../hg/projectx projectx.fi
$ bzr init trunk
$ bzr fast-import projectx.fi trunk
```

#### **Subversion Conversion**

As the most common centralized version control system, migration from Subversion is particularly important for any *new* version control system. Bazaar's svn plugin provides tools for interaction with Subversion projects. In fact, Bazaar can be used transparently with projects stored in Subversion, but that is beyond the scope of this document. (See <a href="http://doc.bazaar-vcs.org/en/migration/foreign/bzr-on-svn-projects.html">http://doc.bazaar-vcs.org/en/migration/foreign/bzr-on-svn-projects.html</a> for more on that subject.) What is relevant here is the <a href="https://svn-import.command-provided">svn-import.command-provided</a> by that plugin. This can import an entire subversion repository including tags and branches, particularly if they are stored in Subversion's recommended directory structure: <a href="https://tags/">/tags/</a>, <a href="https://branches/">/branches/</a> and <a href="https://trunk/">/trunk/</a>.

This command has flexible ways to specify what paths within the Subversion repository contain branches and which contain tags. For example, the recommended layout for Subversion projects (called trunk by the svn plugin) could be specified in ~/.bazaar/subversion.conf as

```
[203ae883-c723-44c9-aabd-cb56e4f81c9a]
branches = branches/*
tags = tags/*
```

This allows substantially complicated Subversion repositories to be converted into a set of separate Bazaar branches. After installing the svn plugin, see bzr help svn-import and bzr help svn-layout.

Home | Documentation | Table of Contents (2.2b1) » Bazaar System previous | next

Administrator's Guide »

# Extending Bazaar with Hooks and Plugins

Bazaar offers a powerful extension mechanism for adding capabilities. In addition to offering full library API access to all of its structures, which can be useful for outside programs that would like to interact with Bazaar branches, Bazaar can also load *plugins* that perform specific tasks. These specific tasks are specified by *hooks* that run during certain steps of the version control process.

For full documentation on the available hooks, see bzr help hooks. Among those, some of the most significant hooks from an administration standpoint are pre\_commit, post\_commit and post\_change\_branch\_tip. A pre\_commit hook can inspect a commit before it happens and cancel it if some criteria are not met. This can be useful for enforcing policies about the code, such as line-endings or whitespace conventions. A post\_commit hook can take actions based on the commit that just happened, such as providing various types of notifications. Finally, a post\_change\_branch\_tip hook is a more general form of a post\_commit hook which is used whenever the tip of a branch changes (which can happen in more ways than just committing). This too can be used for notification purposes, as well as for backups and mirroring.

Information on the whole range of Bazaar plugins is available at <a href="http://doc.bazaar.canonical.com/plugins/en/">http://doc.bazaar.canonical.com/plugins/en/</a>. For purposes of installation, plugins are simply python packages. They can be installed alongside Bazaar in the <a href="bzrlib.plugins">bzrlib.plugins</a> package using each plugin's <a href="setup.py">setup.py</a>. They can also be installed in the plugin path which is the user's <a href="https://www.bazaar/plugins">-/.bazaar/plugins</a> directory or can be specified with the <a href="bzr\_plugin\_path">BZR\_PLUGIN\_PATH</a> environment variable. See <a href="bzr\_help">bzr\_help</a> configuration for more on specifying the location of plugins.

#### **Email Notification**

A common need is for every change made on a branch to send an email message to some address, most often a mailing list. These plugins provide that capability in a number of different ways.

The *email* plugin sends email from each individual developer's computer. This can be useful for situations that want to track what each individual developer is working on. On the downside, it requires that every developer's branches be configured individually to use the same plugin.

The next two plugins *hookless-email* and *email-notifier* address this concern by running on a central server whenever changes happen on centrally stored branches.

#### email

To configure this plugin, simply install the plugin and configure the post\_commit\_to option for each branch. This configuration can be done in the locations.conf file or individually in each branch's branch.conf file. The sender's email address can be specified as post\_commit\_sender if it is different than the email address reported by bzr whoami. The post\_commit\_mailer option specifies how the mail should be sent. If it isn't set, email is sent via /usr/bin/mail. It can also be configured to communicate directly with an SMTP For details on configuring this server. more plugin, http://doc.bazaar-vcs.org/plugins/en/email-plugin.html. As examples, consider the following two possible configurations. A minimal one (USes /usr/bin/mail)

```
[DEFAULT]
post_commit_to = projectx-commits@example.com
```

and a more complicated one (using all of the options)

```
[DEFAULT]
post_commit_url = http://www.example.com/code/projectx/trunk
post_commit_to = projectx-commits@example.com
post_commit_sender = donotreply@example.com
post_commit_mailer = smtplib
smtp_server = mail.example.com:587
smtp_username = bob
# smtp_password = 'not specified, will prompt'
```

#### hookless-email

This plugin is basically a server-side version of the *email* plugin. It is a program that runs either from the command line or as a daemon that monitors the branches specified on the command line for any changes. When a change occurs to any of the monitored branches, it will send an email to the specified address. Using our simple example, the following command would send an email to projectx-commits@example.com on any of the branches under /srv/bzr since the last time the command was run. (This command could be set up to run at regular intervals, for example from cron.)

```
$ bzr_hookless_email.py --email=projectx-commits@example.com \
--recurse /srv/bzr
```

#### email-notifier

This is a more elaborate version of the *hookless-email* plugin that can send templated HTML emails, render wiki-style markup in commit messages and update working copies on the server (similar to push\_and\_update). It can also send emails reporting the creation of new branches or the removal of branches under a specified directory (here /srv/bzr/projectx). As it is more complicated, its configuration is also more complicated and we won't repeat its documentation here, but a simple configuration that will send emails

#### on commits and creation/deletion of branches is

```
[smtp]
server=smtp.example.com
# If user is not provided then no authentication will be perfor
user=bob
password=pAssW0rd
[commits]
# The address to send commit emails to.
to=projctx-commits@example.com
from=$revision.committer
# A Cheetah template used to construct the subject of the email
subject=$relative_path: $revision_number $summary
[new-branches]
to=projectx-commits@example.com
from=donotreply@example.com
subject=$relative_path: New branch created
[removed-branches]
to=projectx-commits@example.com
from=donotreply@example.com
subject=$relative_path: Branch removed
```

If this file is stored as /srv/bzr/email-notifier.conf, then the command

```
$ bzr-email-notifier.py --config=/srv/bzr/email-notifier.conf /
✓
```

will watch all branches under the given directory for commits, branch creations and branch deletions.

## **Feed Generation**

A related concept to sending out emails when branches change is the generation of news feeds from changes on each branch. Interested parties can then choose to follow those news feeds in order to see what is happening on a branch.

#### branchfeed

This plugin creates an ATOM feed for every branch on every branch change (commit, etc.). It stores these files as .bzr/branch/branch.atom inside each branch. Currently, it includes the 20 most recent changes in each feed. To use it, simply install the plugin and set your feed reader to follow the branch.atom files.

In addition, there are other tools that are not plugins for creating news feeds from Bazaar branches. See <a href="http://bazaar-vcs.org/FeedGenerators">http://bazaar-vcs.org/FeedGenerators</a> for more on those tools.

# Mirroring

Sometimes it is useful to ensure that one branch exists as an exact copy of another. This can be used to provide simple backup facilities or redundancy (see Back-up and restore for more details on backups). One way to do this using Bazaar's workflows is to make the branch where changes happen into a bound branch of the mirror branch. Then, when commits happen on the working branch, they will also happen on the mirror branch. Note that commits to bound branches do *not* update the mirror branch's working copy, so if the mirror branch is more than just a backup of the complete history of the branch, for example if it is being served as a web page, then additional plugins are necessary.

## push\_and\_update

This plugin updates Bazaar's push command to also update the remote working copy. It can only work over connections that imply filesystem or SSH access to the remote working copy (bzr+ssh://, sftp:// and file://). Also, it is only useful when the remote branch is updated with an explicit push command.

#### automirror

This plugin is similar to *push\_and\_update* in that it updates the working copy of a remote branch. The difference is that this plugin is designed to update the remote branch on every change to the working branch. To configure this, set the <code>post\_commit\_mirror = URL</code> option on a branch. This option can include multiple branch URLs separated by commas to create multiple mirrors. For example, if we want to mirror our <code>/srv/bzr/projectx/trunk</code> branch to the URL <code>sftp://www.example.com/var/www/projectx</code> (for example if ProjectX

were a web project that we wanted to access at <a href="http://www.example.com/projectx">http://www.example.com/projectx</a>), then we could include

```
[DEFAULT]
post_commit_mirror = sftp://www.example.com/var/www/branches/tr
```

in the file /srv/bzr/projectx/trunk/.bzr/branch/branch.conf.

# Other Useful Plugins

## pqm (plugin)

Facilitating interaction with PQM, this plugin provides support for submitting merge requests to a remote Patch Queue Manager. PQM provides a way to automatically run the test suite before merging changes to the trunk branch.

#### testrunner

Sometimes referred to as the poor man's PQM, this plugin runs a single command on the updated revision (in a temporary directory) and if the command returns 0, then the revision can be committed to that branch. For example, if the testsuite is run with the command nosetests in the root of the branch (which returns 0 if the test suite passes and 1 if it doesn't pass), then one can set

```
[DEFAULT]
pre_change_branch_tip_test_command = nosetests
```

in .bzr/branch/branch.conf.

## checkeol

This plugin is an example of a *pre\_commit* hook that checks the revision being committed for meeting some policy. In this case, it checks that all of the files have the specified line endings. It uses a configuration file <code>.bzreol</code> in the root of the working tree (similar to the <code>.bzrignore</code> file). This configuration file has sections for line feed endings (LF), carriage return/line-feed endings (CRLF) and carriage return endings (CR). For an unusual example that specifies different

line endings for different files, that file might look like

```
[LF]
*.py
*.[ch]

[CRLF]
*.txt
*.ini

[CR]
foo.mac
```

or if you simply want to enforce a single line ending convention on the branch you can use

```
[LF]
*
```

This plugin needs to be installed on the server where the branch updates will happen, and the <code>.bzreol</code> file must be in each branch where line ending policies will be enforced. (Adding it to the branch with <code>bzr add .bzreol</code> is an easy way to ensure this, although it means that branches on the server must have working trees.)

#### text\_checker

This plugin is a more advanced version of *checkeol* that can check such coding style guidelines such as trailing whitespace, long lines and files that don't end with a newline. It is configured using Bazaar's built in rules specification in BZR\_HOME/rules (see bzr help rules for more information. For different types of undesired changes, you can specify different types of actions. For example

```
[name NEWS README]
trailing_whitespace=fail
long_lines=warn
newline_at_eof=ignore
```

```
[name *.py]
tabs=fail
long_line_length=78
long_lines=fail
trailing_whitespace=fail
```

will prevent changes from adding new trailing whitespace to the specified files and keep all python source files free of tabs and lines over 78 characters. To commit while violating these rules, one can pass the --text-check-warn-only option to commit.

Home | Documentation | Table of Contents (2.2b1) » Bazaar System previous | next Administrator's Guide »

Home | Documentation | Table of Contents (2.2b1) » Bazaar System previous | next

Administrator's Guide »

# Web-based code browsing

Browsing the history of a project online is an important part of version control, since it allows people to easily see what happens in a branch without having to have a local, up-to-date copy of that branch. There are a number of possible choices for browsing Bazaar branches on the web, but we will cover one of them in particular detail and briefly mention the other choices where they differ.

# Loggerhead

Loggerhead is a code browsing interface for Bazaar branches (now used in Launchpad). To see an example of Loggerhead in action, browse to <a href="http://bazaar.launchpad.net/~bzr-pqm/bzr/bzr.dev/files">http://bazaar.launchpad.net/~bzr-pqm/bzr/bzr.dev/files</a> which is the loggerhead view of Bazaar's trunk branch. Loggerhead runs as a web application on the server which is accessed over HTTP via a RESTful interface. It is possible to run this application on its own dedicated port as <a href="http://www.example.com/8080">http://www.example.com/8080</a> or to proxy this location behind a separate web server, for example at <a href="http://www.example.com/loggerhead/">http://www.example.com/loggerhead/</a>. We will discuss both of these configurations below.

#### Requirements

Loggerhead depends on a number of other Python packages for the various Web technologies that it builds on. Some of these must be installed to use loggerhead, although some of them are optional. From the loggerhead *README* file, these are

- 1. SimpleTAL for templating. On Ubuntu, *sudo apt-get install python-simpletal* or download from http://www.owlfish.com/software/simpleTAL/download.html
- 2. simplejson for producing JSON data. On Ubuntu, *sudo apt-get install python-simplejson* or use *easy\_install simplejson*.
- 3. Paste for the server. (You need version 1.2 or newer of Paste.) On Ubuntu, *sudo apt-get install python-paste* or use *easy\_install Paste*
- 4. Paste Deploy (optional, needed when proxying through Apache) On Ubuntu, *sudo apt-get install python-pastedeploy* or use *easy\_install PasteDeploy*
- 5. flup (optional, needed to use FastCGI, SCGI or AJP) On Ubuntu, sudo apt-get install python-flup or use easy\_install flup

Although directions for installing these on Ubuntu Linux are given, most other Linux distributions should package these dependencies, making installation easy. For Windows and Mac OS X, they should all be <a href="mailto:easy\_install-able">easy\_install-able</a> or at worst installable from the Python sources.

#### **Built-in Web Server**

Loggerhead has a built-in web server and when started with the serve-branches command, that web server is started on a default port listening on the localhost. If port 8080 (the default) is accessible on www.example.com, then running

```
$ serve-branches --host=www.example.com --port=8080 /srv/bzr
```

will list all of the available branches under that directory on <a href="http://www.example.com:8080/">http://www.example.com:8080/</a>, so that the ProjectX trunk could be browsed at <a href="http://www.example.com:8080/projectx/trunk">http://www.example.com:8080/projectx/trunk</a>. Note that loggerhead provides HTTP access to the underlying Bazaar branches (similar to that described in Smart server over HTTP(S)), so this command should be run as a user without write privileges in /srv/bzr. By default, loggerhead only listens on the localhost, not any external ports, unless specified as above.

## Behind a Proxy

A more common and more safe way to run loggerhead is behind another web server which will proxy certain requests to the loggerhead server on the localhost. To do this, you need to have PasteDeploy installed (see Requirements). Assuming that your server has Apache running, you need to add configuration such as this to set up the proxy

```
ProxyPass http://127.0.0.1:8080/
ProxyPassReverse http://127.0.0.1:8080/
</Location>
```

If your proxy runs at some path within the server, then the servebranches command must be started with the --prefix option. For this example, we could start loggerhead with the command

```
$ serve-branches --prefix=/loggerhead /srv/bzr
```

This would allow the trunk branch of ProjectX to be browsed at http://www.example.com/loggerhead/projectx/trunk.

Loggerhead comes with a script allowing it to run as a service on init.d based Linux systems. Contributions to do a similar thing on Windows servers would be welcomed at http://launchpad.net/loggerhead.

## Other web interfaces

There are a number of other web interfaces available for Bazaar branches (see the list at <a href="http://bazaar-vcs.org/WebInterfaces">http://bazaar-vcs.org/WebInterfaces</a>) and we will just mention a couple of them here for their advantages in particular situations.

#### trac+bzr (http://launchpad.net/trac-bzr)

Trac is a popular web app that integrates a browser for branches, an issue tracker and a wiki. trac+bzr is a trac extension that allows for the trac to be used with Bazaar.

#### webbzr (http://thoughts.enseed.com/webbzr)

This is a notable solution because it is written in pure PHP for web hosts that don't provide a way to run arbitrary Python applications such as Trac or Loggerhead.

#### Redmine (http://redmine.org/)

Like trac, Redmine is a full project management application using the Ruby on Rails framework. It includes support for Bazaar branches.

Home | Documentation | Table of Contents (2.2b1) » Bazaar System previ

previous | next

Home | Documentation | Table of Contents (2.2b1) » Bazaar System previous | next

Administrator's Guide »

# Integration with Other Tools

## Patch Queue Manager (PQM)

## **Bug Trackers**

## **Continuous Integration Tools**

## **Bundle Buggy**

Home | Documentation | Table of Contents (2.2b1) » Bazaar System previous | next Administrator's Guide »

Home | Documentation | Table of Contents (2.2b1) » Bazaar System previous | next

Administrator's Guide »

# Security

#### Authentication

Bazaar's philosophy on authentication is that it is best to reuse existing authentication technologies, rather than trying to reinvent potentially complicated methods for securely identifying users. As such, we describe two such uses of existing software for authentication purposes.

#### **Using SSH**

SSH is a very well tested and featureful technology for authenticating users. For situations where all of the developers have local accounts on the server, it is trivial to provide secure, authenticated bzr+ssh://access. One concern with this method is that it may not be desirable to grant shell access to developers on the server machine. In this case, Bazaar provides bzr\_ssh\_path\_limiter, a script that runs the Bazaar smart server on the server machine at a specified path, and allows no other access.

To set it up, specify:

```
command="/path/to/bzr_ssh_path_limiter <path>" <typical key lin</pre>
```

in each user's ~/.ssh/authorized\_keys file on the server, where <path> is the path to limit access to (and its subdirectories). For more documentation on the syntax of the authorized\_keys file see the documentation of the SSH server. This will only permit Bazaar access to the specified path and no other SSH access for that user.

If it isn't desired to give each user an account on the server, multiple private/public key pairs can be included under one single SSH account (say sshuser) in the ~sshuser/.ssh/authorized\_keys file and

then each developer can be given their own private key. They can then use bzr+ssh://sshuser@server.example.com/ URLs to access the server.

## Using HTTP authentication methods

#### **Access Control**

Many projects need fine-grained access control on who may read and write to which branches. Incorporating these controls into OS-level user accounts using groups and filesystem permissions can be difficult or even not permitted in some instances. Bazaar provides a script called bzr\_access that can be used to provide access control based on usernames, with authentication performed by SSH. To do so, we need to set up private-key authentication in SSH. This can be done using a single SSH user on the server, or one account per user. The idea is to use the SSH's authorized\_keys file to specify the bzr\_access script as the only command that can be run by a user identified by a particular key pair.

First, you will need to generate a private/public key pair for each user who will be accessing the repository. The private key should be distributed to the user and the public key will be needed on the server to identify the user. On the server, in the SSH user's ~/.ssh/authorized\_keys file, use the following line for each repository user and the corresponding public key:

```
command="/path/to/bzr_access /path/to/bzr /path/to/repository <
```

where <key> is the (possibly very long) public key, <type> is the type of SSH key and <username> is the username to associate with that public key.

The bzr\_access script obtains its configuration information from the file /path/to/repository/bzr\_access.conf. This file should not be placed under version control in branch located a at /path/to/repository since that would allow anyone with access to the repository to change the control rules. The access

bzr\_access.conf file is in a simple INI-style format with sections defined by [groups] and [/]. The options in the [groups] section are the names of groups and the values of those options should be the usernames in that group. Inside the [/] section, the options are usernames or group names (prefixed with @) and the values are either rw, r or nothing, representing read-write access, read-only access or no access at all. A sample of bzr\_access.conf could be:

```
[groups]
admins = alpha
devels = beta, gamma, delta

[/]
@admins = rw
@devels = r
upsilon =
```

where the user whose key is associated with *alpha* would have readwrite access, the users *beta*, *gamma* and *delta* would have read-only access and user *upsilon* would not be able to access any branches under /path/to/repository.

#### Additional Considerations with bzr access

As currently written, bzr\_access only allows each public key to be associated with a single repository location. This means that if developers need to access two or more different repositories, then each developer will need to have two or more private keys for SSH and be able to select between them (see man ssh for more information on configuring multiple private keys).

Also, each repository can only have a single configuration file, with access configured for all branches in the repository. This means that if different access rules are needed for different projects, then those projects must be in different repositories. This then necessitates the

use of multiple private keys as just described.

Finally, as noted above under Using SSH all of the public keys may be included in the authorized\_keys file of a single user on the server. It is also possible to use a single private/public key pair for all of the developers, but this only allows a single username for access control to the repository (since the username is associated with the public key in authorized\_keys. While this is certainly possible it seems to defeat the purpose of fine-grained access control, although it does provide the same limited SSH access as that described above.

Home | Documentation | Table of Contents (2.2b1) » Bazaar System previous | next Administrator's Guide »

Home | Documentation | Table of Contents (2.2b1) » Bazaar System previous | next

Administrator's Guide »

## Back-up and Restore

Backing up Bazaar branches can be done in two different ways. If an existing filesystem-based backup scheme already exists, then it can easily be used where the Bazaar branches reside. Alternately, Bazaar itself can be used to mirror the desired branches to or from another location for backup purposes.

## Filesystem Backups

Bazaar transactions are atomic in the sense that the disk format is such that it is in a valid state at any instant in time. However, for a backup process that takes a finite amount of time to complete, it is possible to have inconsistencies between different on-disk structures when backing up a live branch or repository. (Bazaar itself manages this concurrency issue by only *reading* those structures in a well-defined order.) Tools such as LVM that allow instantaneous snapshots of the contents of a disk can be used to take filesystem backups of live Bazaar branches and repositories.

For other backup methods, it is necessary to take the branch or repository offline while the backup is being done in order to guarantee consistency between the various files that comprise a Bazaar branch's history. This requirement can be alleviated by using Bazaar as its own backup client, since it follows an order for reading that is designed to manage concurrent access (see the next section for details). Depending on the different access methods that are being used for a branch, there are different ways to take the branch "offline". For bzr+ssh:// access, it is possible to temporarily change the filesystem permissions to prevent write access from any users. For http:// access, changing permissions, shutting down the HTTP server or switching the server to a separate configuration that disallows access are all possible ways to take a branch offline for backup. Finally, for direct filesystem access, it is necessary to make the branch directories un-writable.

Because this sort of downtime can be very disruptive, we strongly encourage using Bazaar itself as a backup client, where branches are copied and updated using Bazaar directly.

### Bazaar as its own backup

The features that make Bazaar a good distributed version control system also make it a good choice for backing itself up. In particular, complete and consistent copies of any branch can easily be obtained with the branch and pull commands. As a result, a backup process can simply run bzr pull on a copy of the main branch to fully update that copy. If this backup process runs periodically, then the backups will be as current as the last time that pull was run. (This is in addition to the fact that revisions are immutable in Bazaar so that a prior revision of a branch is always recoverable from that branch when the revision id is known.)

As an example, consider a separate backup server that stores backups in /var/backup. On that server, we could initially run

```
$ cd /var/backup
$ bzr branch bzr+ssh://server.example.com/srv/bzr/trunk
$ bzr branch bzr+ssh://server.example.com/srv/bzr/feature-gui
```

to create the branches on the backup server. Then, we could regularly (for example from cron) do

```
$ cd /var/backup/trunk
$ bzr pull # the location to pull from is remembered
$ cd ../var/backup/feature-gui
$ bzr pull # again, the parent location is remembered
```

The action of pulling from the parent for all branches in some directory is common enough that there is a plugin to do it. The bzrtools plugin contains a multi-pull command that does a pull in all branches under a specified directory.

With this plugin installed, the above periodically run commands

#### would be

```
$ cd /var/backup
$ bzr multi-pull
```

Note that multi-pull does a pull in *every* branch in the specified directory (the current directory by default) and care should be taken that this is the desired effect. A simple script could also substitute for the multi-pull command while also offering greater flexibility.

#### **Bound Branch Backups**

When bzr pull is run regularly to keep a backup copy up to date, then it is possible that there are new revisions in the original branch that have not yet been pulled into the backup branch. To alleviate this problem, we can set the branches up so that new revisions are *pushed* to the backup rather than periodically pulling. One way to do this is using Bazaar's concept of bound branches, where a commit in one branch happens only when the same commit succeeds in the branch to which it is *bound*. As a push-type technology, it is set up on the server itself rather than on the backup machine. For each branch that should be backed up, you just need to use the bind command to set the URL for the backup branch. In our example, we first need to create the branches on the backup server (we'll use bzr push, but we could as easily have used bzr branch from the backup server)

```
$ cd /srv/bzr/projectx/trunk
$ bzr push bzr+ssh://backup.example.com/var/backup/trunk
$ cd ../feature-gui
$ bzr push bzr+ssh://backup.example.com/var/backup/feature-gui
```

and then we need to bind the main branches to their backups

```
$ cd ../trunk
$ bzr bind bzr+ssh://backup.example.com/var/backup/trunk
```

```
$ cd ../feature-gui
$ bzr bind bzr+ssh://backup.example.com/var/backup/feature-gui
```

A branch can only be bound to a single location, so multiple backups cannot be created using this method.

Using the automirror plugin mentioned under Hooks and Plugins, one can also make a push-type backup system that more naturally handles multiple backups. Simply set the post\_commit\_mirror option to multiple URLs separated by commas. In order to backup to the backup server and a remote location, one could do

```
$ cd /srv/bzr/trunk
$ echo "post_commit_mirror=bzr+ssh://backup.example.com/var/bac
bzr+ssh://offsite.example.org/projectx-corp/backup/trunk" >> .b
$ cd ../feature-gui
$ echo "post_commit_mirror=bzr+ssh://backup.example.com/var/bac
bzr+ssh://offsite.example.org/projectx-corp/backup/feature-gui"
```

As for any push-type backup strategy that maintains consistency, the downside of this method is that all of the backup commits must succeed before the initial commit can succeed. If there is a many mirror branches or if the bound branch has a slow network connection, then the delay in the original commit may be unacceptably long. In this case, pull-type backups, or a mixed system may be preferable.

## Restoring from Backups

#### Checking backup consistency

Many a system administrator has been bitten by having a backup process, but when it came time to restore from backups, finding out that the backups themselves were flawed. As such, it is important to check the quality of the backups periodically. In Bazaar, there are two ways to do this: using the bzr check command and by simply making a new branch from the backup. The bzr check command goes through all of the revisions in a branch and checks them for validity according to Bazaar's internal invariants. Since it goes through every revision, it can be quite slow for large branches. The other way to ensure that the backups can be restored from is to perform a test restoration. This means performing the restoration process in a temporary directory. After the restoration process, bzr check may again be relevant for testing the validity of the restored branches. The following two sections present two restoration recipes.

#### Restoring Filesystem Backups

There are many different backup tools with different ways of accessing the backup data, so we can't cover them all here. What we will say is that restoring the contents of the <code>/srv/bzr</code> directory completely will restore all branches stored there to their state at the time of the backup (see Filesystem Backups for concerns on backing up live branches.) For example, if the backups were mounted at <code>/mnt/backup/bzr</code> then we could restore using simply:

```
$ cd /srv
$ mv bzr bzr.old
$ cp -r /mnt/backup/bzr bzr
```

Of course, to restore only a single branch from backup, it is sufficient to copy only that branch. Until the restored backup has been successfully used in practice, we recommend keeping the original directory available.

### Restoring Bazaar-based Backups

In order to restore from backup branches, we can simply branch them into the appropriate location:

```
$ cd /srv
$ mv bzr bzr.old
$ cd bzr
$ bzr branch bzr+ssh://backup.example.com/var/backup/trunk
$ bzr branch bzr+ssh://backup.example.com/var/backup/feature-gu
```

If there are multiple backups, then change the URL above to restore from the other backups.

Home | Documentation | Table of Contents (2.2b1) » Bazaar System previous | next

Administrator's Guide »

Home | Documentation | Table of Contents (2.2b1) » Bazaar System previous | next

Administrator's Guide »

## **Upgrades**

Bazaar has a strong commitment to inter-version compatibility both on disk and over the network. Newer clients should be able to interact with older versions on the server (although perhaps not at optimal speed) and older clients should also be able to communicate with newer versions of Bazaar on the server. Divergences from this rule are considered bugs and are fixed in later versions.

That said, Bazaar is constantly improving and the most recent versions are the most featureful and have better performance. In particular, the Bazaar versions 2.0 and later have significant advantages over earlier versions, including a more compact disk faster network operations and overall performance improvements. With the 2.0 release, Bazaar has moved to a stable/development release model where the 2.x series maintained with months. bugfixes releases for Six simultaneously the 2.(x+1) series is being developed with monthly beta releases that are suitable for everyday use. development has a stable trunk with an extensive test suite, so there is no reason to fear using the development series for everyday use, it simply changes more often than the stable series. Many users do run the development version of Bazaar and update it regularly, including most of the Bazaar developers themselves.

## Software upgrades

Upgrading the Bazaar software is as simple as re-installing the Python package using either the latest binary package for Windows or Mac OS X, the binary package provided by your Linux distribution, or installing from the source release. See <a href="http://bazaar-vcs.org/Downloads">http://bazaar-vcs.org/Downloads</a> for the latest releases for all supported platforms.

Bazaar's later versions support all of the earlier disk formats (back to the very first one), so there is no need to upgrade the branches on the disk when upgrading the software. To make use of particular new features that might need updated versions on both the server and developer's machines, it does not matter if the clients or the servers are upgraded first.

### Disk format upgrades

In its evolution, Bazaar has used a sequence of disk formats for improved storage efficiency and speed. With the new disk format released in version 2.0, there is a commitment to keep that disk format until version 3.0 is released, which has not even been planned yet. (Bazaar 2.0 was released almost two years after Bazaar 1.0.) As a result, disk format upgrades should be extremely infrequent.

If there are existing branches in an older format that you would like to upgrade to the latest format, you can see the 2.0 Upgrade Guide for more information. From the system administration perspective, it is important to coordinate the timing of various upgrades in the process. First, the central branches on the server should be upgraded. Next, any local mirrors that developers have should be upgraded. Finally, developers' local branches should be upgraded. These upgrades will require an appropriate version of the software whenever they are performed. (It is possible to upgrade branches remotely over the network, but it may be much slower.)

## Plugin upgrades

When Bazaar does update its version, plugins that use the Bazaar API may need to be upgraded to reflect changes in that API. Some plugins have strict version dependencies on the version of the Bazaar API that they will accept. If this is the case, then you should ensure that the plugins you depend on have been updated *before* you upgrade your Bazaar version to avoid a situation where your plugins won't work with the installed version of Bazaar. If this does happen, then the solution is simply to reinstall the previous version of Bazaar that *did* work with the plugins. For installations that depend on a large number of plugins, this sort of version upgrade should be tested in a safe sandbox to ensure that the entire collection of Bazaar and its plugins can all work together.

Home | Documentation | Table of Contents (2.2b1) » Bazaar System Administrator's Guide »

previous | next

Home | Documentation | Table of Contents (2.2b1) » Bazaar System Administrator's Guide »

previous

# **Advanced Topics**

# System Monitoring

## **Capacity Planning Tips**

## Clustering

### Multi-site Setups

The "distributed" in distributed version control system should indicate that Bazaar is well suited for multi-site development situations and indeed, that is the case. The advantage comes from the ease and transparency of managing merges between branches with divergent history. Note that there are many, many different ways to manage widely-flung development setups using Bazaar and its branching and merging capabilities. These can be discovered and tested before being implemented as policy. We will describe one such possible setup here.

Consider ProjectX Corp's international expansion with a new branch office in Darwin, Australia, in addition to the company's headquarters in Austin, Texas, USA. One of the difficulties of a far-flung multi-site development environment such as this is that the network connection between Australia and Texas is slow and unreliable. So, each branch office would like the master branch to be local to them. (In situations with good network connectivity, a local branch bound to the remote master may be all that is needed to support multi-site development.)

Of course, with two master branches, there is always the question of which one is authoritative. Given Bazaar's facility at managing multiple branches, we suggest that it is best not to privilege either the Texas or Australia branches, but to merge both of them into a separate master branch (which may reside at either site). For definiteness, we will locate the master branch at the Texas site. So, we will have three branches stored on two servers: trunk and texasintegration at the Texas site and australia-integration at the Darwin site. These branches are named in terms of the sites where the development takes place, but in many cases it may make more sense to name branches after the functional teams rather their

geographical locations. Since we are trying illustrate the issues with multi-*site* development, we will persist in this naming scheme.

#### Setup

Using our previous setup at the Texas site, we will simply rename the old trunk branch as trunk and branch a copy as texas-integration.

In Australia, we need to set up the /srv/bzr/projectx directory and get a copy of the current trunk as australia-integration:

```
$ mkdir -p /srv/bzr
$ cd /srv/bzr
$ bzr init-repo --no-trees projectx
$ cd projectx
$ bzr branch bzr+ssh://server.example.com/srv/bzr/trunk
$ bzr branch trunk australia-integration
```

#### Merging to master

Then, each office works with their local copy of the trunk. At some point, sooner or later depending on the pace of development in the two locations, the two local trunks need to be merged. (In general, sooner beats later when merging, since there is no penalty for multiple merges.) In this example, Alice at the Texas office will do the merging on her local machine using branches on the server:

```
# Get a copy of the Australia branch in Texas. After the initi
# command, use pull to keep the branch up to date. With a slow
# this is the only slow part
$ bzr branch bzr+ssh://autralia.example.com/srv/bzr/projectx/au
bzr+ssh://server.example.com/srv/bzr/projectx/australia-integ
```

Note that Bazaar does not commit even cleanly applied merges by default. This is because although a merge may apply cleanly, the merged state still needs to be checked before it is committed. (Just because there are no text conflicts does not mean that everything will work after a merge.) An alternative that can pull when possible and merge otherwise is available with bzr merge --pull.

### Merging back to local trunks

Now the trunk branch is the most up-to-date version of the software and both of the local trunks need to reincorporate the changes from the master. If no new commits have been made to texas-integration, then that can happen using bzr pull:

If new changes have happened on texas-integration since the integration with trunk, then the above pull will produce an error suggesting to use merge:

```
$ bzr merge ../trunk
# Run test suite, resolve conflicts
$ bzr commit -m "Merging Australian changes"
```

In Australia, they will need to update their local copy of trunk:

```
$ cd /srv/bzr/projectx/trunk
$ bzr pull # parent location is used by default
```

Then, they need to pull or merge the changes from trunk into the local trunk. This should be done by a developer with a checkout of australia-integration so that they can run the test suite:

```
$ cd ~/projectx
$ bzr co bzr+ssh://australia.example.com/srv/bzr/projectx/austr
$ cd australia-integration
$ bzr merge bzr+ssh://australia.example.com/srv/bzr/projectx/tr
# Run test suite and integrate Texan changes with only recent l
# development
$ bzr commit -m "Integrate work from Texas"
```

#### Other Considerations

Multi-site deployments can be complicated, due to the many possible variations of development velocity, divisions of labor, network connectivity, resources for integration, etc. The preceding description is meant to be one possible way to do fairly symmetric multi-site development. (Neither Texas or Australia is privileged in this structure.) In a situation where there is one main site and other smaller sites, one of the local trunk branches can be eliminated and trunk can be used directly for development at the main site.

It is also up to the particular situation how frequently the local trunks are integrated into the master trunk. Given resources specifically for integration, it is conceivable that a developer may be constantly responsible for integrating changes from the two teams.

Alternatively, the two sites could work on well-separated, well-defined features and merge to the master trunk only when their respective features are complete. Given the difficulty of resolving conflicts in very large merges and the ease of merge handling in Bazaar, we suggest that merges be done more frequently, rather than less.

Home | Documentation | Table of Contents (2.2b1) » Bazaar System

Administrator's Guide »

previous

previous | next

# Bazaar User Guide

## Introduction

- Introducing Bazaar
- Core concepts
- Workflows

## Getting started

- Installing Bazaar
- Entering commands
- Getting help
- Configuring Bazaar
- Using aliases
- Using plugins
- Bazaar Zen

## Personal version control

- Going solo
- Starting a project
- Controlling file registration
- Reviewing changes
- Recording changes
- Browsing history
- Releasing a project
- Undoing mistakes

## Sharing with peers

- Working with another
- Branching a project
- Merging changes
- Resolving conflicts
- Annotating changes

## Team collaboration, central style

- Centralized development
- Publishing a branch
- Using checkouts
- Working offline on a central branch
- Reusing a checkout

## Team collaboration, distributed style

- Distributed development
- Organizing branches
- Using gatekeepers
- Sending changes

## Miscellaneous topics

- The journey ahead
- Pseudo merging
- Shelving Changes
- Filtered views
- Using stacked branches
- Running a smart server
- Using hooks
- Exporting version information

# A brief tour of some popular plugins

- BzrTools
- bzr-svn

# Integrating Bazaar into your environment

- Web browsing
- Bug trackers

## **Appendices**

- Specifying revisions
- Organizing your workspace
- Advanced shared repository layouts
- Configuring email
- Serving Bazaar with Apache
- Writing a plugin

Home | Documentation | Table of Contents (2.2b1) »

previous | next

## **Tutorials**

- Bazaar in five minutes
- Bazaar Tutorial
- Using Bazaar with Launchpad
- Centralized Workflow Tutorial

Home | Documentation | Table of Contents (2.2b1) »

previous | next

# Quick Reference

- PDF format
- PNG format
- SVG format

Home | Documentation | Table of Contents (2.2b1) »

previous | next

## **Bazaar Release Notes**

#### **List of Releases**

- Bazaar Release Notes
- bzr 2.2b1
- bzr 2.1.1
- bzr 2.1.0
- bzr 2.1.0rc2
- bzr 2.1.0rc1
- bzr 2.0.5
- bzr 2.0.4
- bzr 2.1.0b4
- bzr 2.0.3
- bzr 2.1.0b3
- bzr 2.1.0b2
- bzr 2.0.2
- bzr 2.1.0b1
- bzr 2.0.1
- bzr 2.0.0
- bzr 2.0.0rc2
- bzr 2.0.0rc1
- bzr 1.18.1
- bzr 1.18
- bzr 1.18rc1
- bzr 1.17.1 (unreleased)
- bzr 1.17
- bzr 1.16.1
- bzr 1.16
- bzr 1.15
- bzr 1.14

- bzr 1.13
- bzr 1.12
- bzr 1.11
- bzr 1.11rc1
- bzr 1.10
- bzr 1.10rc1
- bzr 1.9
- bzr 1.9rc1
- bzr 1.8
- bzr 1.8rc1
- bzr 1.7.1
- bzr 1.7.1rc1
- bzr 1.7
- bzr 1.7rc2
- bzr 1.7rc1
- bzr 1.6.1
- bzr 1.6.1rc2
- bzr 1.6.1rc1
- bzr 1.6
- bzr 1.6rc5
- bzr 1.6rc4
- bzr 1.6rc3
- bzr 1.6rc2
- bzr 1.6rc1
- bzr 1.6beta3
- bzr 1.6beta2
- bzr 1.6beta1
- bzr 1.5
- bzr 1.5rc1
- bzr 1.4
- bzr 1.4rc2
- bzr 1.4rc1
- bzr 1.3.1
- bzr 1.3.1rc1

- bzr 1.3
- bzr 1.3rc1
- bzr 1.2
- bzr 1.2rc1
- bzr 1.1
- bzr 1.1rc1
- bzr 1.0
- bzr 1.0rc3
- bzr 1.0rc2
- bzr 1.0rc1
- bzr 0.92
- bzr 0.92rc1
- bzr 0.91
- bzr 0.91rc2
- bzr 0.91rc1
- bzr 0.90
- bzr 0.90rc1
- bzr 0.18
- bzr 0.18rc1
- bzr 0.17
- bzr 0.17rc1
- bzr 0.16
- bzr 0.16rc2
- bzr 0.16rc1
- bzr 0.15
- bzr 0.15rc3
- bzr 0.15rc2
- bzr 0.15rc1
- bzr 0.14
- bzr 0.14rc1
- bzr 0.13
- bzr 0.13rc1
- bzr 0.12
- bzr 0.12rc1

- bzr 0.11
- bzr 0.11rc2
- bzr 0.11rc1
- bzr 0.10
- bzr 0.9.0
- bzr 0.8.2
- bzr 0.8.1
- bzr 0.8
- bzr 0.7
- bzr 0.6
- bzr 0.1.1
- bzr 0.1
- bzr 0.0.9
- bzr 0.0.8
- bzr-0.0.7
- bzr-0.0.6
- bzr-0.0.5
- bzr-0.0.4
- bzr-0.0.3
- bzr-0.0.2.1
- bzr-0.0.2
- bzr-0.0.1
- bzr-0.0.0.69

# Bazaar 2.0 Upgrade Guide

#### Overview

#### High level upgrade process

In broad terms, there are 3 steps involved in upgrading to Bazaar 2.x:

- 1. Upgrade the core software
- 2. Upgrade required plugins
- 3. Migrate data to the new default format.

Bazaar 2.x supports branches in earlier formats so the third step is strictly not required. However, the new default format in Bazaar 2.x is more space efficient, faster on large projects and provides a range of new features, so it is recommended that most projects migrate to it at a convenient time.

For most users, upgrading to 2.x and migrating to the new format is straight forward. For projects with a large community of developers though, things become more complex. In these cases, careful planning and good communications become essential. This document provides general advice which aims to assist in this regard. If in doubt, please contact us on our mailing list or IRC channel with any questions or concerns you have.

### Upgrading the core software

The steps required to upgrade the core software vary from operating system to operating system. There is nothing special about upgrading from Bazaar 1.x to Bazaar 2.0 compared to upgrading from Bazaar 1.x to Bazaar 1.y. In either case, a brief outline of the steps is given below.

#### To upgrade Bazaar on Linux:

- 1. Ensure your package manager is configured with the required software sources, e.g. the official release PPA for Ubuntu: https://launchpad.net/~bzr/+archive
- 2. Use your package manager to upgrade to the latest version.

#### To upgrade Bazaar on Windows:

- 1. Uninstall the existing version using Add/Remove Programs.
- 2. Install the new version using the relevant installer.

To upgrade Bazaar on OS X (via the installer):

1. Install the new version using the relevant installer.

To upgrade Bazaar on OS X (via MacPorts):

- 1. Refresh the package metadata using sudo port selfupdate
- 2. Upgrade to the latest version using **sudo port upgrade bzr**

For further information on installing and upgrading, see http://bazaar-vcs.org/Download.

## Upgrading required plugins

Many plugins are not dependent on a particular Bazaar version so upgrading them is optional. Other plugins, notably bzrtools and bzrsvn, are more tightly associated with Bazaar's APIs so these typically need to be upgraded in lockstep with the core software.

For Windows and OS X users, bzrtools and bzr-svn are typically included in the installer so no special steps are required to upgrade these. For Linux and UNIX users, bztrools, bzr-svn and many other popular plugins can be installed and upgraded using your platform's package manager, e.g. Synaptic on Ubuntu.

## Migrating data to the new default format

As mentioned earlier, the complexity of migrating to a new format depends on several factors, particularly project community size. It also depends on how data is currently stored, e.g. in a standalone branch, multiple branches in a shared repository, stacked branches on Launchpad, etc. These various scenarios are covered in the next chapter.

## Data migration

#### Preparing for data migration

Before starting a migration, there are a few important things to do first:

- 1. Take a complete backup.
- 2. Take some time to purge obsolete branches.

A complete backup gives you a safety net in case anything goes wrong.

Purging obsolete branches reduces the amount of data that needs to be migrated. See Finding obsolete branches later for some tips on doing this.

#### Introducing the upgrade-related commands

There are 3 important commands to be aware of when migrating data.

- check check a repository, branch or tree for data integrity errors
- reconcile fix data integrity errors
- upgrade migrate data to a different format.

**reconcile** is rarely needed but it's good practice to run **check** before and after running **upgrade**.

For detailed help on these commands, see the Bazaar User Reference.

#### Communicating with your community

To enable a smooth transition to the new format, you should:

- 1. Make one person responsible for migrating the trunk.
- 2. Test the migration of trunk works successfully.
- 3. Schedule a time for the trunk migration and notify your community in advance.

This advance warning should be long enough for users to have time to upgrade Bazaar and any required plugins before the migration date.

For larger projects, allow some time for the migration itself. You should have a good idea of how long the migration will take after doing the test migration. It may make sense to do the migration on a weekend or a Friday, giving yourself some breathing space if things go wrong.

After the trunk is migrated, you'll need to notify your community accordingly, giving them instructions as to how to migrate their local branches. Sample instructions are provided later in this document.

### Migrating a standalone branch

#### The steps are:

- 1. Run bzr check.
- 2. If there are errors, try using **bzr reconcile** to fix them. If that fails, file a bug so we can help you resolve the issue and get your trunk clean. If it works, take a backup copy of your now clean trunk.
- 2. Run bzr upgrade -format where format is 2a or later.
- 3. Run **bzr check** to confirm the final result is good.

### Migrating branches in a shared repository

Upgrade things in the following order:

- 1. Upgrade the shared repository.
- 2. Upgrade the branches.
- 3. Upgrade any lightweight checkouts.

As in the standalone branch case, be sure to run **check** before and after the upgrade to check for any existing or introduced issues.

#### Migrating branches on Launchpad

You have two options for upgrading your Launchpad branches. You can either upgrade them remotely or you can upgrade them locally and push the migrated branch to Launchpad. We recommend the latter. Upgrading remotely currently requires a fast, rock solid network connection to the Launchpad servers, and any interruption in that connection can leave you with a partially upgraded branch. The instructions below are the safest and often fastest way to upgrade your Launchpad branches.

To allow isolation between public and private branches, Launchpad uses stacked branches rather than shared repositories as the core technology for efficient branch storage. The process for migrating to a new format for projects using Launchpad code hosting is therefore different to migrating a personal or in-house project.

In Launchpad, a project can define a *development series* and associate a branch with that series. The branch then becomes the *focus of development* and gets special treatment and a shortcut url. By default, if anybody branches your project's focus of development and pushes changes back to Launchpad, their branch will be stacked on your development focus branch. Also, branches can be associated with other Launchpad artifacts such as bugs and merge proposals. All of these things mean that upgrading your focus of

development branch is trickier.

Here are the steps to follow:

- 1. The nominated person grabs a copy of trunk and does the migration locally.
- 2. On Launchpad, unset the current trunk from being the development focus. (This *must* be done or the following step won't work as expected.)
  - 1. Go to your project's home page on Launchpad
  - 2. Look for "XXX is the current focus of development"
  - 3. Click on the edit (pencil) icon
  - 4. Click on "Change details" in the portlet on the right
  - 5. Scroll down to where it says "Branch: (Optional)"
  - 6. Blank out this input field and click "Change"
- 3. Push the migrated trunk to Launchpad. See below if you want your new migrated development focus branch to have the same name as your old pre-migration development focus branch.
- 4. Set it as the development focus. Follow the instructions above but at step 5, enter the name of the newly migrated branch you just pushed.
- 5. Ask users subscribed to the old trunk to subscribe to the new one.

In summary, these steps mean that the old trunk is still available and existing branches stacked on it will continue to be so. However, the development focus has switched to the migrated trunk and any new branches pushed to Launchpad for your project will now stack on it.

You are now ready to tell your community that the new trunk is available and to give them instructions on migrating any local branches they have.

If you want your new migrated development focus branch to have the same name as your old pre-migration branch, you need to do a few extra things before you establish the new development focus.

- Rename your old pre-migration branch; use something like fooobsolete-do-not-use. You will really not want to delete this because there will be artifacts (bugs, merge proposals, etc.) associated with it.
- 2. Rename the new migrated branch to the pre-migration branch's old name.
- 3. Re-establish the development focus branch using the new migrated branch's new name (i.e. the old pre-migration branch's original name).

# Migrating local branches after a central trunk has migrated

#### To migrate a standalone branch:

- 1. Grab the latest branch from the central location into a new directory.
- 2. Pull or merge any changes you've made in your existing branch into the new branch.

#### To migrate branches in a shared repository:

- 1. Create a fresh shared repository in the new format (2a or later).
- 2. Grab the latest branch from the central location into a new directory inside the shared repository.
- 3. Decide which of your local branches you want to migrate. (If you haven't already, now's a good time for Finding obsolete branches and purging them, after backing up first of course.)
- 4. To migrate each local branch of interest, there are 2 options:
  - **init** an empty branch in the new repository and **pull** the revisions from the branch in the old repository across.
  - In the new repository, **branch** from trunk to the new branch

name then **merge** your changes from the matching branch in the old repository.

The first method will give you a branch which is identical (in terms of revision history) to the old branch, but it's parent branch will be set to the old branch, not your new trunk. If you use this method, you'll probably want to edit your branch.conf file to update the parent branch setting.

In contrast, the second approach sets up the parent branch correctly. However, it isn't ideal if you're not ready to include all the latest revisions from trunk into that branch yet.

## Tips and tricks

#### Finding obsolete branches

If you use feature branching for developing each fix and enhancement separately, you may have several old branches that are no longer required. In many cases, the relevant changes may now be merged into trunk. In other cases, a branch may be obsolete thanks to another change made by yourself or others.

When checking for an obsolete branch, there are three things in particular to confirm:

- 1. The working tree has no in-progress changes.
- 2. The working tree has no shelved changes.
- 3. Any locally committed revisions have been merged into the parent branch.

After changing into the root of a branch, the commands to check these things respectively are:

```
bzr status
bzr shelve --list
bzr missing --mine-only
```

If your branches are stored in a shared repository locally, you may find the *Local Changes* tab in Bazaar Explorer's repository view helpful here (revision 159 or later) as it shows a summary of these things, excluding the shelve information currently, for each branch as you select it.

## Bazaar User Reference

### **About This Manual**

This manual is generated from Bazaar's online help. To use the online help system, try the following commands.

Introduction including a list of commonly used commands:

```
bzr help
```

List of topics and a summary of each:

```
bzr help topics
```

List of commands and a summary of each:

```
bzr help commands
```

More information about a particular topic or command:

```
bzr help topic-or-command-name
```

The following web sites provide further information on Bazaar:

Home page: http://bazaar.canonical.com/
Official docs: http://doc.bazaar.canonical.com/

**Launchpad:** https://launchpad.net/bzr/

## Concepts

- Branches
- Checkouts
- Content Filters
- Criss-Cross
- Diverged Branches
- End of Line Conversion
- Storage Formats
- Patterns
- Repositories
- Rules
- Standalone Trees
- Branches Out of Sync
- Working Trees

#### Lists

- Authentication Settings
- Bug Tracker Settings
- Configuration Settings
- Conflict Types
- Current Storage Formats
- Debug Flags
- Environment Variables
- Files
- Global Options
- Hooks
- Location aliases
- Log Formats
- Other Storage Formats
- Revision Identifiers
- Standard Options
- Status Flags
- URL Identifiers

# Commands

Home | Documentation | Table of Contents (2.2b1) »

previous | next

# Bazaar System Administrator's Guide

- Introduction
  - Scope of this guide
  - What you need to run a Bazaar server
- Simple Setups
  - Smart server
- Other Setups
  - Dumb servers
  - Smart server over HTTP(S)
  - Direct Smart Server Access
- Migration
  - Fast Import
  - Subversion Conversion
- Extending Bazaar with Hooks and Plugins
  - Email Notification
  - Feed Generation
  - Mirroring
  - Other Useful Plugins
- Web-based code browsing
  - Loggerhead
  - Other web interfaces
- Integration with Other Tools
  - Patch Queue Manager (PQM)
  - Bug Trackers
  - Continuous Integration Tools
  - Bundle Buggy
- Security
  - Authentication
  - Access Control
- Back-up and Restore

- Filesystem Backups
- Bazaar as its own backup
- Restoring from Backups
- Upgrades
  - Software upgrades
  - Disk format upgrades
  - Plugin upgrades
- Advanced Topics
  - System Monitoring
  - Capacity Planning Tips
  - Clustering
  - Multi-site Setups

Home | Documentation | Table of Contents (2.2b1) »

previous | next

# What's New in Bazaar 2.1?

This document outlines the major improvements in Bazaar 2.1 vs Bazaar 2.0. As well as summarizing improvements made to the core product, it highlights enhancements within the broader Bazaar world of potential interest to those upgrading.

Bazaar 2.1.0 marks the start of our second long-term-stable series. This series will be supported with bug fixes for the next 12 months. All users are encouraged to upgrade from the 2.0.x stable series.

# Better efficiency

Many operations now consume less memory. Several operations are also faster including branching, logging merged revisions and upgrading from pre-2a to 2a format.

# New command options

Several commands have new options. These include:

Command	Option	Description
branch	bind	Bind to the source location
commit	commit-time	Give an explicit commit timestamp
switch	revision	Switch to a particular revision
unshelve	keep	Apply changes but don't delete them
unshelve	preview	Show the diff that would result from unshelving
update	revision	Update to a particular revision

Other command improvements include:

- A .../user-reference/shelve-help editor can be defined to provide shelf functionality at a granularity finer than per-patch hunk.
- ../user-reference/send-help send now supports the OS X Mail application.

See the help for the commands above for further details.

# Per-file merge hooks

Hooks can now be defined for smart merging of selected file types. This enables easier merging of ChangeLogs, Release Notes and other file that frequently conflict.

#### **DWIM** revision identifiers

Revision identifiers can now be given in a *Do-What-I-Mean* style. For example, you can now just give a tag (instead of saying tag:xxx) or just say today (instead of saying date:today). Prefixes are now only required if the revision spec could be ambiguous.

# New ignore patterns

Patterns prefixed with I are exceptions to ignore patterns and take precedence over regular ignores. Such exceptions are used to specify files that should be versioned which would otherwise be ignored. Patterns prefixed with II act as regular ignore patterns, but have highest precedence, even over the I exception patterns.

#### Smart server home directory support

bzr+ssh and bzr paths can now be relative to home directories specified in the URL. Paths starting with a path segment of ~ are relative to the home directory of the user running the server, and paths starting with ~user are relative to the home directory of the named user. For example, for a user "bob" with a home directory of /home/bob, these URLs are all equivalent:

- bzr+ssh://bob@host/~/repo
- bzr+ssh://bob@host/~bob/repo
- bzr+ssh://bob@host/home/bob/repo

If bzr serve was invoked with a --directory argument, then no home directories outside that directory will be accessible via this method.

This is a feature of bzr serve, so pre-2.1 clients will automatically benefit from this feature when bzr on the server is upgraded.

# Generalized glob support on Windows

On Windows, glob expansion is now handled in a universal way across all commands. In previous versions, it was explicitly handed by just a few commands, e.g. add. A side effect of this change is that patterns now need to be quoted when passed to the ignore command, e.g. bzr ignore \*.foo now needs to be given as bzr ignore "\*.foo".

# Improved Git and Mercurial interoperability

Many improvements have been made to the git and hg plugins. With these plugins installed, most Git and Mercurial repositories can now be read by standard Bazaar clients. Changes can also be written back via the dpush command.

#### Metaprojects

New plugins are available for constructing larger projects from smaller ones. These include:

- builder construction of a branch using recipes
- externals Subversion-style external branches

**Note:** The builder plugin has been designed to complement the builddeb plugin to streamline Debian source package management. It may also be useful for building test images for a QA team or disk images for installers, say.

# Colocated branch workspaces

A colocated workspace is one where a single working tree is used across one or more branches managed at that same location. This is now supported by the new colo plugin and by Bazaar Explorer.

#### Better documentation

A *Bazaar System Administrator's Guide* covering topics such as setting up servers, security, backups and email integration has been added.

A large number of documentation bugs have been fixed, clarifying numerous issues and filling in some missing holes.

The *Bazaar User Reference* has been organized into topics making it easier to navigate through and print selected sections of.

To assist users migrating from other tools, a Survival Guide has been published explaining Bazaar to users of other tools in terms they already know. Sections are provided for existing users of CVS, Subversion, ClearCase, Perforce, Visual SourceSafe, Git, Mercurial, Darcs and Monotone.

Selected documents have also been translated to Japanese.

#### **Enhanced GUI clients**

Numerous enhancements have been made to most of our GUIs including Bazaar Explorer, TortoiseBZR and the QBzr-Eclipse addon. These applications all build on top of improvements made to QBzr. Bzr-gtk has also been improved.

Bazaar Explorer has over a dozen new features including smart toolbars, support for all bzr commands (including those in plugins), a better working tree browser and a submit delta report showing the cumulative effect of a series of commits. See What's New in Bazaar Explorer 1.0? for more information.

#### **Further information**

For more detailed information on the changes made, be sure to check the *Bazaar Release Notes* for:

- the interim bzr milestones
- the plugins you use.

Enjoy, The Bazaar Development Team

Home | Documentation | Table of Contents (2.2b1) »

# Bazaar in five minutes

#### Introduction

Bazaar is a distributed version control system that makes it easier for people to work together on software projects.

Over the next five minutes, you'll learn how to put your files under version control, how to record changes to them, examine your work, publish it and send your work for merger into a project's trunk.

If you'd prefer a more detailed introduction, take a look at Learning More.

#### Installation

This guide doesn't describe how to install Bazaar but it's usually very easy. You can find installation instructions at:

- **GNU/Linux:** Bazaar is probably in your GNU/Linux distribution already.
- Windows: installation instructions for Windows.
- Mac OS X: installation instructions for Mac OS X.

For other platforms and to install from source code, see the Download and Installation pages.

#### Introducing yourself

Before you start working, it is good to tell Bazaar who you are. That way your work is properly identified in revision logs.

Using your name and email address, instead of John Doe's, type:

```
$ bzr whoami "John Doe <john.doe@gmail.com>"
```

Bazaar will now create or modify a configuration file, including your name and email address.

Now, check that your name and email address are correctly registered:

```
$ bzr whoami
John Doe <john.doe@gmail.com>
```

#### Putting files under version control

Let's create a directory and some files to use with Bazaar:

```
$ mkdir myproject
$ cd myproject
$ mkdir subdirectory
$ touch test1.txt test2.txt test3.txt subdirectory/test4.txt
```

**Note for Windows users:** use Windows Explorer to create your directories, then right-click in those directories and select New file to create your files.

Now get Bazaar to initialize itself in your project directory:

```
$ bzr init
```

If it looks like nothing happened, don't worry. Bazaar has created a branch where it will store your files and their revision histories.

The next step is to tell Bazaar which files you want to track. Running bzr add will recursively add everything in the project:

```
$ bzr add
added subdirectory
added test1.txt
added test2.txt
added test3.txt
added subdirectory/test4.txt
```

Next, take a snapshot of your files by committing them to your branch. Add a message to explain why you made the commit:

```
$ bzr commit -m "Initial import"
```

As Bazaar is a distributed version control system, it doesn't need to

connect to a central server to make the commit. Instead, Bazaar stores your branch and all its commits inside the directory you're working with; look for the .bzr sub-directory.

#### Making changes to your files

Let's change a file and commit that change to your branch.

Edit test1.txt in your favourite editor, then check what have you done:

```
$ bzr diff
=== modified file 'test1.txt'
--- test1.txt    2007-10-08 17:56:14 +0000
+++ test1.txt    2007-10-08 17:46:22 +0000
@@ -0,0 +1,1 @@
+test test test
```

Commit your work to the Bazaar branch:

```
$ bzr commit -m "Added first line of text"
Committed revision 2.
```

# Viewing the revision log

You can see the history of your branch by browsing its log:

```
$ bzr log
revno: 2
committer: John Doe <john.doe@gmail.com>
branch nick: myproject
timestamp: Mon 2007-10-08 17:56:14 +0000
message:
   Added first line of text
revno: 1
committer: John Doe <john.doe@gmail.com>
branch nick: myproject
timestamp: Mon 2006-10-08 17:46:22 +0000
message:
   Initial import
```

#### Publishing your branch with sftp

There are a couple of ways to publish your branch. If you already have an SFTP server or are comfortable setting one up, you can publish your branch to it.

Otherwise, skip to the next section to publish with Launchpad, a free hosting service for Bazaar.

Let's assume you want to publish your branch at www.example.com/myproject:

```
$ bzr push --create-prefix sftp://your.name@example.com/~/publi
2 revision(s) pushed.
```

Bazaar will create a myproject directory on the remote server and push your branch to it.

Now anyone can create their own copy of your branch by typing:

```
$ bzr branch http://www.example.com/myproject
```

**Note:** to use sftp, you may need to install paramiko and pycrypto. See http://bazaar-vcs.org/InstallationFaq for details.

#### Publishing your branch with Launchpad

Launchpad is a suite of development and hosting tools for free software projects. You can use it to publish your branch.

If you don't have a Launchpad account, follow the account signup guide and register an SSH key in your new Launchpad account.

Replacing john.doe with your own Launchpad username, type [1]:

```
$ bzr push lp:~john.doe/+junk/myproject
```

[1] Use of the 1p: URL scheme requires bzr 0.92 or later.

**Note:** +junk means that this branch isn't associated with any particular project in Launchpad.

Now, anyone can create their own copy of your branch by typing:

```
$ bzr branch lp:~john.doe/+junk/myproject
```

You can also see information about your branch, including its revision history, at https://code.launchpad.net/people/+me/+junk/myproject

# Creating your own copy of another branch

To work with someone else's code, you can make your own copy of their branch. Let's take a real-world example, Bazaar's GTK interface:

```
$ bzr branch lp:~bzr/bzr-gtk/trunk bzr-gtk.john
Branched 292 revision(s).
```

Bazaar will download all the files and complete revision history from the bzr-gtk project's trunk branch and create a copy called bzrgtk.john.

Now, you have your own copy of the branch and can commit changes with or without a net connection. You can share your branch at any time by publishing it and, if the bzr-gtk team want to use your work, Bazaar makes it easy for them to merge your branch back into their trunk branch.

#### Updating your branch from the main branch

While you commit changes to your branch, it's likely that other people will also continue to commit code to the parent branch.

To make sure your branch stays up to date, you should merge changes from the parent into your personal branch:

```
$ bzr merge
Merging from saved parent location: http://bazaar.launchpad.net
All changes applied successfully.
✓
```

Check what has changed:

```
$ bzr diff
```

If you're happy with the changes, you can commit them to your personal branch:

```
$ bzr commit -m "Merge from main branch"
Committed revision 295.
```

#### Merging your work into the parent branch

After you've worked on your personal branch of bzr-gtk, you may want to send your changes back upstream to the project. The easiest way is to use a merge directive.

A merge directive is a machine-readable request to perform a particular merge. It usually contains a patch preview of the merge and either contains the necessary revisions, or provides a branch where they can be found.

Replacing mycode.patch, create your merge directive:

```
$ bzr send -o mycode.patch
Using saved parent location: http://bazaar.launchpad.net/~bzr/b
```

You can now email the merge directive to the bzr-gtk project who, if they choose, can use it merge your work back into the parent branch.

# Learning more

You can find out more about Bazaar in the Bazaar User Guide.

To learn about Bazaar on the command-line:

```
$ bzr help
```

To learn about Bazaar commands:

```
$ bzr help commands
```

To learn about the "foo" topic or command:

```
$ bzr help foo
```

Home | Documentation | Table of Contents (2.2b1) » Tutorials »

previous | next