# What's New

This What's New is associated with the 2007-08-29 release of Amazon EC2. This guide was last updated on March 10, 2008.

The following table describes the important changes since the last release of the Amazon EC2 Developer Guide.

Change	Description	Release Date
DevPay	Amazon DevPay officially released. Amazon DevPay documentation changes were made throughout the guide. For more information, see <a href="Creating Paid">Creating Paid</a> <a href="AMIs">AMIs</a> .	18 December 2007
New Instance Types	In addition to the standard Amazon EC2 instance, we now provide large (m1.large) and extra-large (m1.xlarge) instances. For more information, see Selecting Instance Types.	20 October 2007
Instance Launch Time	RunInstances and DescribeInstances now return the time when the Amazon EC2 instance was launched. For more information, see <a href="ec2-run-instances">ec2-run-instances</a> and <a href="ec2-describe-instances">ec2-describe-instances</a> .	20 October 2007

## Welcome

#### **Topics**

- Audience
- How This Guide Is Organized
- Related Resources

This is the *Amazon Elastic Compute Cloud* Developer Guide. This section describes who should read this guide, how the guide is organized, and other resources related to Amazon Elastic Compute Cloud.

The Amazon Elastic Compute Cloud will occasionally be referred to within this guide as simply "Amazon EC2"; all copyrights and legal protections still apply.

#### **Audience**

This guide picks up where the Getting Started Guide ends and provides you with the information to create more sophisticated AMIs, using advanced service features.

#### **Required Knowledge and Skills**

Use of this guide assumes you are familiar with the following:

- XML (See <u>W3 Schools XML Tutorial</u>)
- Basic understanding of web services (See <u>W3 Schools Web Services</u> <u>Tutorial</u>))
- A programming language for consuming a web service and any related tools

You should also have worked through the *Amazon EC2 Getting Started Guide*, installed the command line and API tools, and have a general understanding of the service.

### **How This Guide Is Organized**

This guide is organized into several major sections described in the table below.

Information	Relevant Sections
Describes how to create a customized software package (operating system and applications) that you can run on Amazon EC2	<ul> <li>Creating and <u>Preparing AMIs</u></li> </ul>
Describes Amazon EC2 instances and provides tips for using them effectively	• Launching and Using Instances
Describes instance network addressing, explains the distributed firewall, and provides usage examples	<ul> <li>Instance Addressing and Network Security</li> </ul>
Explains the basics of using the SOAP and Query APIs, including signing requests	Using the APIs
Provides a comprehensive reference to the SOAP and Query APIs	API Reference
Provides a comprehensive reference to the Amazon EC2 command line tools	• <u>Command Line</u> <u>Tools Reference</u>
Provides answers to commonly asked questions	<u>Technical FAQ</u>
Describes Amazon EC2 terms	• Glossary
Typographic and symbol conventions	• Document Conventions

Each section is written to stand on its own, so you should be able to look up the information you need and go back to work. However, you can also read through the major sections sequentially to get in-depth knowledge about Amazon EC2.

# **Related Resources**

The table below lists related resources that you'll find useful as you work with this service.

Resource	Description		
Amazon EC2 Getting Started Guide	The Getting Started Guide provides a quick tutorial of the service based on a simple use case. Examples and instructions are included.		
<u>Amazon EC2 Release Notes</u>	The Release Notes give a high-level overview of the current release. They specifically note any new features, corrections, and known issues.		
AWS Developer Resource Center	A central starting point to find documentation, code samples, release notes, and other information to help you build innovative applications with AWS.		
Amazon EC2 product information	The primary web page for information about Amazon EC2.		
Discussion Forums	A community-based forum for developers to discuss technical questions related to Amazon Web Services.		
E-mail address for questions related to your AWS account: < <u>webservices@amazon.com</u> >	This e-mail address is <i>only</i> for account questions. For technical questions, use the Discussion Forums.		
Conditions of Use	Detailed information about the copyright and trademark usage at Amazon.com.		

# Introduction to Amazon Elastic Compute Cloud

Amazon EC2 is a web service that enables you to allows you to provision ondemand compute capacity from Amazon's world class data centers.

This means you can allocate or release resources within minutes, not hours or weeks, as your application requires. Most importantly, you only pay for what you use. If you use a server instance for an hour, you pay for an hour. If you use an instance for a year, you pay for a year.

Although the applications for Amazon EC2 are only limited by your ingenuity, there are popular ways in which our customers use Amazon EC2:

- **Scalable Applications**—you can build a scalable application that shrinks or expands to meet your current demands. This can help you use only the compute resources that you need and can help you respond to events where a mention on a popular news site can result in a dramatic spike in traffic.
- **Temporary Events**—you can use Amazon EC2 for temporary solutions and one-off events that would require you to maintain a fleet of compute resources that are normally idle. This includes hosting conferences in virtual worlds, live blogging, distribution of newly released media, and short-term promotional websites.
- **Batch Processing**—you can use Amazon EC2 for projects that require massive compute resources which would be expensive to build on your own. This includes video and image processing, financial data processing, and science and research applications.

# **Creating and Preparing AMIs**

#### **Topics**

- Creating an AMI
- Bundling an AMI
- Building Shared AMIs
- Sharing AMIs
- Paid AMIs

This section details how to build, store, and share AMIs.

# **Creating an AMI**

There are two common ways to create an AMI that offer a mix of ease of use and detailed customization levels.

The easiest method involves starting from an existing public AMI and modifying it according to your requirements, as described in <u>Starting with an Existing AMI</u>.

Another approach is to build a fresh installation either on a stand-alone machine or on an empty file system mounted by loopback. This essentially entails building an operating system installation from scratch and is described in <a href="Creating an AMI through a Loopback File">Creating an AMI through a Loopback File</a>.

After the installation package has been built to your satisfaction, you must bundle it and upload it to Amazon Simple Storage Service (Amazon S3) as described in <u>Bundling an AMI</u>.

# **Starting with an Existing AMI**

To quickly and easily get a new working AMI, start with an existing public AMI or one of your own. You can then modify it and create a new AMI with the ec2-bundle-vol utility decribed in <a href="Bundling an AMI">Bundling an AMI</a>.



Before selecting an AMI, determine whether the instance types you plan to launch are 32-bit or 64-bit. For more information, see <u>Instance Types</u>

Make sure you are using GNU Tar 1.15 or later.

To use an existing AMI to create a new AMI:

- 1. Select an AMI
- 2. Generate a Key Pair
- 3. Launch the Instance
- 4. Authorize Network Access
- 5. Connect to the Instance
- 6. <u>Upload the Key and Certificate</u>

#### **Select an AMI**

First, locate an AMI that contains the packages and services you require. This can be one of your own AMIs or a public AMIs provided by Amazon EC2.

#### To select an AMI:

1. Get a list of available AMIs by entering the <a href="ec2-describe-images">ec2-describe-images</a> command:

```
PROMPT> ec2-describe-images -a
```

The response includes the image ID, the location of the file in Amazon S3, and whether the file is available.

2. Choose an AMI from the list and write down its AMI ID.

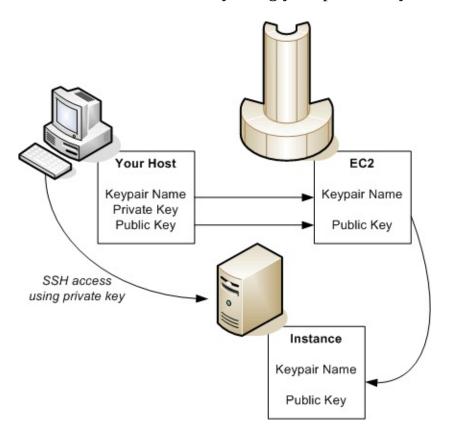
#### **Example**

```
PROMPT> ec2-describe-images -o self -o amazon
IMAGE ami-60a54009 ec2-public-images/base-fc4-apache.manifest.xml 475219
IMAGE ami-61a54028 <your-s3-bucket>/image.manifest.xml 495219933132 avail
IMAGE ami-2bb65342 ec2-public-images/getting-started.manifest.xml 475219
IMAGE ami-6ea54007 ec2-public-images/base-fc3-mysql.manifest.xml 4752198
```

#### Generate a Key Pair

This step is only required if you selected one of the public AMIs provided by Amazon EC2. You must create a public/private key pair to ensure that only you have access to instances that you launch.

After you generate a key pair, the public key is stored in Amazon EC2 using the key pair name you selected. Whenever you launch an instance using the key pair name, the public key is copied to the instance metadata. This allows you to access the instance securely using your private key.



#### To create a public/private key pair

1. Enter the following command:

```
PROMPT> ec2-add-keypair < keypair-name>
```

where <keypair-name> is the name you select for the key pair.

The resulting private key is displayed.

- 2. Open a text editor.
- 3. Paste the entire private key, starting with the line "----BEGIN RSA PRIVATE KEY-----" and ending with the line "----END RSA PRIVATE KEY-----".
- 4. Save the file and exit.



This file should only be readable by the file owner.

#### **Example**

```
PROMPT> ec2-add-keypair gsg-keypair
KEYPAIR gsg-keypair 1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:1
----BEGIN RSA PRIVATE KEY----
MIIEoQIBAAKCAQBuLFq5ujHrtm1jnutSuo08Xe56LlT+HM8v/xkaa39EstM3/aFxTHqElQi.
HungXQ29VTc8rc1bW0lkdi230H5eqkMHGhvEwqa0HWASUMll4o3o/IX+0f2UcPoKC0VUR+j>
5AU52EQfanIn3ZQ8lFW7Edp5a3q4DhjGlUKToHVbicL5E+g45zfB95wIyywWZfeW/UUF3Lp(
ebIUla1aTbHkLbCC2r7RTn8vp0Wp47BGVYGtGSBMpTRP5hnbzzuai3itkiLHiU39S2sJCJ07
i8BygR4s3mHKBj8l+ePQxG1kGbF6R4yq6sECmXn17MRQVXODNHZbAqMBAAECqqEAY1tsiUs]
91CXirkYGuVfLyLflXenxfI50mDFms/mumTqloH07tr0oriHDR5K7wMcY/YY5YkcXNo7mvU\
ZNUJs7rw9gZRTrf7LylaJ58k0cyajw8TsC4e4LPbFaHwS1d6K8rXh64o6WgW4SrsB6ICmr1l
3wcfgt5ecIu4TZf00E9IHjn+2eRlsrjBde0Ri7KiUNC/pAG23I6MdD0FEQRcCSigCj+4/mci
SWS4dMbrpb9FNSIcf9dcLxVM7/6KxgJNfZc9XWzUw77Jg8x92Zd0fVhH0ux5IZC+UvSKWB4c
tE8C3p9bbU9VGyY5vLCAiIb4qQKBgQDLi024GXrIkswF32YtBBMuVgLGCwU9h9Hl09mKAc2r
jUE5IpzRjTedc9I2qiIMUTwtqnw42auSCzbUeYMURPtDqyQ7p6AjMujp9EPemcSV0K9vXYL(
xW9MC0dtV6iPkCN7g0giZXPRKaFbWADp16p8UAIvS/a5XXk5jwKBgQCKkpHi2EISh1uRkhx1
iDCiK6JBRsMvpLbc0v5dKwP5alo1fmdR5PJaV2qvZSj5CYNpMAv1/EDNTY50SIJU+0KFm0bv
rdLNLDL4+TcnT7c62/aH01ohYaf/VCbRhtLlBfgGoQc7+sAc8vmKkesnF7CgCEKDyF/dhrx\
gC0iZzzNAapayz1+JcVTwwEid6j9JgNXbBc+Z2YwMi+T0Fv/P/hwkX/ype0XnIUcw0Ih/Yt(
DQbsz7LcY1HqXiHKYNWNvXgww0+oiChjxvEkSdsTTIfnK4VSCvU9BxDbQHjdiNDJbL6oar92
rBYvChJZF7LvUH4YmVpHAoGAbZ2X7XvoeE0+uZ58/BGK0IGHByHBDiXtzMhdJr15HTYjxK7(
gK+8zp4L9IbvLGDMJ08vft32XPEWuvI8twCzFH+CsWLQADZMZKSsBas0Z/h1FwhdMgCMcY+(
JZKjTSu3i7vhvx6RzdSedXEMNTZWN4qlIx3kR5aHcukCqYA9T+Zrvm1F0seQPbLknn7EqhX]
P8TTvW/6bdPi23ExzxZn7KOdrfclYRph1LHMpAONv/x2xALIf91UB+v5ohy1oDoasL0gij1k
2ERKKdwz0ZL9SWg6VTdhr/5G994CK72fy5WhyERbDjUIdHaK3M849JJuf8cSrvSb4g==
----END RSA PRIVATE KEY----
```

#### Launch the Instance

You are now ready to launch an instance of the AMI that you previously selected.

#### To launch an instance

1. Start the launch by entering the following command:

```
PROMPT> ec2-run-instances <ami_id> - K <keypair-name>
```

where <ami\_id> is the AMI ID you selected earlier and <keypair-name> is the name of the key pair. The command will return the AMI instance ID, a unique identifier for each launched instance. You use the instance ID to manipulate the instance. This includes viewing the status of the instance, terminating the instance, and so on.

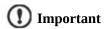
Launching the instance will take a few minutes.

2. View the progress of the instance by entering the following command:

```
PROMPT> ec2-describe-instances <instance_id>
```

where <instance\_id> is the ID of the instance.

When the status field displays "running," the instance was created and is booting. However, the instance might not be immediately accessible over the network. Make sure to use the appropriate DNS name provided by the ec2-describe-instances command.



Once you launch an instance, you will be billed for all usage, including hourly CPU time. Make sure to terminate any instances that you do not want to leave running. For information on Amazon EC2 pricing, refer to the <a href="Manazon EC2">Amazon EC2</a> home page.

#### **Example**

The following example launches an instance of ami-2bb65342.

```
PROMPT> ec2-run-instances ami-2bb65342 -k gsg-keypair
RESERVATION r-302dc059 416161254515 default
INSTANCE i-eb977f82 ami-2bb65342 pending
```

The following shows the status of the launch:

### **Authorize Network Access**

To reach a running instance from the Internet, you must enable access for the SSH service on port 22.

#### To enable SSH service on port 22

• Enter the following command:

PROMPT> ec2-auth	orize <b>defa</b> l	ult -p 22					
PERMISSION	default	ALLOWS tcp	22	22	FROM	CIDR	(

#### **Connect to the Instance**

After starting an instance, you can log in and modify it according to your requirements.

#### To connect to an instance

• If you are launching an AMI that supports SSH login (e.g., public AMIs), use the following command to log in with your private key:

```
PROMPT> ssh -i root@<dns_location>
```

where *<private-keyfile*> is the file that contains the private key and *dns\_location* is the DNS location of the instance within Amazon EC2. Your instance displays a prompt that contains your username and the hostname of the instance.

You now have complete control over the instance. You can add, remove, modify, or upgrade packages and files to suit your needs.



We recommend exercising extreme care when changing some of the basic Amazon EC2 configuration settings, such as the network interface configuration and the /etc/fstab contents. Otherwise, the AMI might become unbootable or inaccessible from the network once running.

#### Example

The following example shows logging in to an AMI using SSH.

```
PROMPT> ssh -i id_rsa-gsg-keypair root@domU-12-34-31-00-00-05.usma1.compute.amazor root@domU-12-34-31-00-00-05 #
```

#### **Upload the Key and Certificate**

Your new AMI is encrypted and signed to ensure that only you and Amazon EC2 can access it. Therefore, you must upload your Amazon EC2 private key and X.509 certificate to the running instance, for use in the AMI bundling process.



For information on obtaining your Amazon EC2 private key and X.509 certificate, refer to the Amazon Elastic Compute Cloud Getting Started Guide.

#### To upload your Amazon EC2 private key and X.509 certificate

- 1. Copy your Amazon EC2 private key and X.509 certificate to the /mnt directory.
- 2. Enter the following command:

```
PROMPT> scp <private_keyfile> <certificate_file> root@<dns_location>:/mnt
```

where *<private\_keyfile*> is the file that contains the private key, certificate file is the file that contains the certificate, and dns location is the DNS location of the instance within Amazon EC2.

Amazon EC2 returns the name of the files and some performance statistics.



It is important that the key and cert files are uploaded into /mnt to prevent them from being bundled with the new AMI.

You are ready to bundle the volume and uploading the resulting AMI to Amazon S3. For more information, see Bundling an AMI.

#### **Example**

# Creating an AMI through a **Loopback File**

This method involves doing a full operating system installation on a clean root file system, but avoids having to create a new root disk partition and file system on a physical disk. Once you have installed your operating system, the resulting image can be bundled as an AMI with the ec2-bundle-image utility.



Before selecting an AMI, determine whether the instance types you plan to launch are 32-bit or 64-bit. For more information, see Instance Types Make sure you are using GNU Tar 1.15 or later.

These examples use Fedora Core 4. Please make any adjustments for your distribution.

#### **AMI Creation Process**

Create a File to Host the AMI 1 Create a Root File System Inside the File 2 Mount the File through Loopback 3 Prepare for the Installation 5 **Install the Operating System Configure the Operating System** 

#### Create a File to Host the AMI

The dd utility can create files of arbitrary sizes. Make sure to create a file large enough to host the operating system, tools, and applications that you will install. For example, a baseline Linux installation requires about 700MB, so your file should be at least 1 GB.

#### To create a file to host the AMI

• Enter the following command:

```
# dd if=/dev/zero of=<image_name> bs=1M =<size>
```

where <image\_name> is the name of the image file you are creating and <size> is the size of the file in megabytes.

#### **Example**

The following command creates a one gigabyte file (1024\*1MB):

```
# dd if=/dev/zero of=my-image.fs bs=1M count=1024
1024+0 records in
1024+0 records out
```

#### **Create a Root File System Inside the File**

There are several variations on the mkfs utility that can create a file system inside the image file you are creating. Typical Linux installations default to ext2 or ext3 file systems.

#### To create an ext3 file system

• Enter the following command:

```
# mke2fs -F -j <image_name>
```

where <image\_name> is the name of the image file.

#### **Example**

The following command creates an ext3 file system:

```
# mke2fs -F -j my-image.fs
mke2fs 1.38 (30-Jun-2005)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
131072 inodes, 262144 blocks
13107 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=268435456
8 block groups
32768 blocks per group, 32768 fragments per group
16384 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done
This filesystem will be automatically checked every 24 mounts or
180 days, whichever comes first. Use tune2fs -c or -i to override.
```

#### **Mount the File through Loopback**

The loopback module allows you to use a normal file as if it were a raw device, which gives you a file system within a file. Mounting a file system image file through loopback presents it as part of the normal file system. You can then modify it using your favorite file management tools and utilities.

#### To mount the file through loopback

1. Create a mount point in the file system where the image will be attached:

```
# mkdir <image_mountpoint>
```

where <image\_mountpoint> is the location where the image will be mounted.

2. Mount the file system image:

```
# mount -o loop <image_name> <image_mountpoint>
```

where <image\_name> is the name of the image file and <image\_mountpoint> is the mount location.

#### **Example**

The following commands create and mount the my-image.fs image file:

```
# mkdir /mnt/ec2-fs
# mount -o loop my-image.fs /mnt/ec2-fs
```

#### **Prepare for the Installation**

Before the operating system installation can proceed, you must create and prepare the newly created root file system.

#### To prepare for the installation

1. Create a /dev directory and populate it with a minimal set of devices (you can ignore the errors in the output):

```
# mkdir /mnt/ec2-fs/dev
# /sbin/MAKEDEV -d <image_mountpoint>/dev -x console
# /sbin/MAKEDEV -d <image_mountpoint>/dev -x null
# /sbin/MAKEDEV -d <image_mountpoint>/dev -x zero
```

where <image\_mountpoint> is the mount location.

2. Create the fstab file within the /etc directory and add the following:

```
/dev/sda1 /
                           defaults
                                          1 1
                   ext3
none
          /dev/pts devpts
                          gid=5, mode=620 0 0
          /dev/shm tmpfs
                           defaults
                                          0 0
none
          /proc
                   proc
                           defaults
                                          0 0
none
          /sys
                   sysfs
                           defaults
                                          0 0
none
```

3. Create a temporary yum configuration file (e.g., yum-xen.conf) and add the following components:

```
[main]
cachedir=/var/cache/yum
debuglevel=2
logfile=/var/log/yum.log
exclude=*-debuginfo
gpgcheck=0
obsoletes=1
reposdir=/dev/null

[base]
name=Fedora Core 4 - $basearch - Base
```

```
mirrorlist=http://fedora.redhat.com/download/mirrors/fedora-core-$r
enabled=1

[updates-released]
name=Fedora Core 4 - $basearch - Released Updates
mirrorlist=http://fedora.redhat.com/download/mirrors/updates-releas
enabled=1
```

This ensures all the required basic packages and utilities are installed. This file can be located anywhere on your main file system (not on your loopback file system) and is only used during installation.

#### 4. Enter the following:

```
# mkdir <image_mountpoint>/proc
# mount -t proc none <image_mountpoint>/proc
```

where <image\_mountpoint> is the mount location. A groupadd utility bug in the shadow-utils package (versions prior to 4.0.7-7) requires you to mount the new proc file system manually with the above command.

#### **Example**

These commands create the /dev directory and populate it with a minimal set of devices:

```
# mkdir /mnt/ec2-fs/dev
# /sbin/MAKEDEV -d /mnt/ec2-fs/dev -x console
MAKEDEV: mkdir: File exists
MAKEDEV: mkdir: File exists
MAKEDEV: mkdir: File exists
# /sbin/MAKEDEV -d /mnt/ec2-fs/dev -x null
MAKEDEV: mkdir: File exists
MAKEDEV: mkdir: File exists
MAKEDEV: mkdir: File exists
MAKEDEV: mkdir: File exists
# /sbin/MAKEDEV -d /mnt/ec2-fs/dev -x zero
MAKEDEV: mkdir: File exists
MAKEDEV: mkdir: File exists
MAKEDEV: mkdir: File exists
MAKEDEV: mkdir: File exists
```

This example creates and mounts the /mnt/ec2-fs/proc directory

# mkdir /mnt/ec2-fs/proc
# mount -t proc none /mnt/ec2-fs/proc

#### **Install the Operating System**

At this stage, the basic directories and files are created and you are ready to install the operating system. Depending on the speed of the host and network link to the repository, this process might take a while.

#### To install the operating system

• Enter the following command:

```
# yum -c <yum_configuration_file> --installroot=<<u>image_mountpoint</u>> -y groupin:
```

where <yum\_configuration\_file> is the name of the yum configuration file and <image\_mountpoint> is the mount location.

You now have a base installation, which you can configure for operation inside Amazon EC2 and customize for your use.

#### Example

This example installs the operating system at the /mnt/ec2-fs mount point using the yum-xen.conf yum configuration file.

```
# yum -c yum-xen.conf --installroot=/mnt/ec2-fs -y groupinstall Base
Setting up Group Process
Setting up repositories
base
                  100% |=======|
                                                00
updates-released
                  100% |========|
                                         1.1 kB
                                                00
comps.xml
                  100% |====== | 693 kB
                                                00
comps.xml
                  100% |====== | 693 kB
                                                00
Setting up repositories
Reading repository metadata in from local files
                  100% |====== | 824 kB
                                                00
primary.xml.gz
       Added 2772 new packages, deleted 0 old in 15.32 seconds
primary.xml.gz
                  100% |======= | 824 kB
Added 2772 new packages, deleted 0 old in 10.74 seconds
Complete!
```

#### **Configure the Operating System**

After successfully installing the base operating system, you must configure the networking and hard drives to work in the Amazon EC2 environment.

#### To configure the operating system

1. Edit (or create) /mnt/ec2-fs/etc/sysconfig/network-scripts/ifcfgetho and make sure it contains at least the following information:

```
DEVICE=eth0
B00TPR0T0=dhcp
ONBOOT=ves
TYPE=Ethernet
USERCTL=yes
PEERDNS=yes
IPV6INIT=no
```



The Amazon EC2 DHCP server ignores hostname requests. If you set DHCP HOSTNAME, the local hostname will be set on the instance but not externally. Additionally, the local hostname will be the same for all instances of the AMI, which might be confusing.

2. Ensure that networking starts by making sure the following line appears in the /mnt/ec2-fs/etc/sysconfig/network file:

```
NETWORKING=yes
```

3. Amazon EC2 provides the instance with additional local disk storage on /dev/sda2 and swap space on /dev/sda3. To ensure both these are mounted at system startup, add the following lines to /mnt/ec2fs/etc/fstab:

/dev/sda2 /mnt ext3 defaults /dev/sda3 swap swap defaults	0 0 0 0	
--	------------	--



The /dev/sda2 and /dev/sda3 storage locations only apply to small instances. For more information on instance storage, see <a href="the section called "Instance Storage">the section called "Instance Storage"</a>.

4. Make sure all of your required services start at system startup by allocating them appropriate system run levels. For example, to enable the service myservice on multi-user and networked run levels, enter the following commands:

```
# chroot /mnt/ec2-fs /bin/sh
# chkconfig --level 345 my-service on
# exit
```

5. After your new installation is successfully installed and configured to operate in the Amazon EC2 environment, you can unmount the image:

```
# umount <image_mountpoint>/proc
# umount -d <image_mountpoint>
```

where <image\_mountpoint> is the mount location.

#### **Example**

The following example unmounts the installation from the /mnt/ec2-fs mount point.

```
# umount /mnt/ec2-fs/proc
# umount -d /mnt/ec2-fs
```

# **Bundling an AMI**

To use a file system image with Amazon EC2, you must bundle it as an AMI. The bundling process does the following:

- Compresses the image to minimize bandwidth usage and storage requirements.
- Encrypts and signs the compressed image to ensure confidentiality and authenticates the image against its creator.
- Splits the encrypted image into manageable parts for upload.
- Creates a manifest file that contains a list of the image parts with their checksums.

This section describes the AMI tools that automate this process and provides examples of their use.

The AMI tools include three command-line utilities:

- ullet ec2-bundle-image bundles an existing AMI
- ec2-bundle-vol creates an AMI from an existing machine or installed volume
- ec2-upload-bundle uploads a bundled AMI to Amazon S3 storage

#### **Installing the AMI Tools**

The AMI tools are packaged as an RPM suitable for running on Fedora Core with Ruby 1.8.2 (or greater) installed. You need root privileges to install the software.

The AMI tools RPM is available from our <u>public Amazon S3 downloads bucket</u>.

#### To install the AMI tools

1. Install Ruby using the yum package manager:

```
# yum install ruby
```

2. Install the AMI tools RPM:

```
# rpm -i ec2-ami-tools-x.x-xxxx.i386.rpm
```

### **Installation Issues**

The AMI tools libraries install in /usr/lib/site\_ruby.

If you receive a load error when running one of the AMI utilities, Ruby might not have found the path. To fix this, add /usr/lib/site\_ruby to Ruby's library path, which is set in the RUBYLIB environment variable.

# **Viewing Documentation**

#### To view the manual for each utility

• Append --manual to the command that invokes the utility:

```
# ec2-bundle-image --manual
```

#### To view help for each utility

• Append --help to the command that invokes the utility:

```
# ec2-bundle-image --help
```

#### **Bundling an AMI Using the AMI Tools**

After creating a machine image, it must be bundled as an AMI for use with Amazon EC2. How you bundle the image depends on how you created the image (for information about creating AMIs, see <u>Creating an AMI</u>).

#### To bundle the image you prepared in a loopback file

• Enter the following command:

```
# ec2-bundle-image -i <image_name>.img -k <private_keyfile> -C <certificate_file>
```

where <image\_name> is the name of the image file, <private\_keyfile> is the file that contains the private key, <certificate\_file> is the file that contains the certificate, and <user\_id> is the user ID associated with your account.



The user ID is your AWS account ID without dashes. It is the same as your Amazon Access ID and consists of 12 digits.

# To bundle an AMI you created by taking a snapshot of the local machine root file system (requires root privileges)

• Enter the following command:

```
# ec2-bundle-vol -k <private_keyfile> -C <certificate_file> -U <user_id>
```

where <private\_keyfile> is the file that contains the private key, <certificate\_file> is the file that contains the certificate, and <user\_id> is the user ID associated with your account.



Make sure to disable SELinux when running ec2-bundle-vol.



The user ID is your AWS account ID without dashes. It is the same as your Amazon Access ID and consists of 12 digits.

#### **Example**

This command bundles an image created in a loopback file:

```
# ec2-bundle-image -i my-image.fs -k pk-HKZYKTAIG2ECMXYIBH3HXV4ZBZQ55CLO.pem -C cert-HKZ
image.part.00
image.part.01
...
image.part.NN
image.manifest.xml
```

This command bundles the local machine root file system:

```
# ec2-bundle-vol -k pk-HKZYKTAIG2ECMXYIBH3HXV4ZBZQ55CL0.pem -C cert-HKZYKTAIG2ECMXYIBH3HX image.part.00 image.part.01 ... image.part.NN image.manifest.xml
```

#### **Uploading a Bundled AMI**

You must upload the bundled AMI to Amazon S3 before it can be accessed by Amazon EC2. Use ec2-upload-bundle to upload the bundled AMI that you created earlier. Amazon S3 stores data objects in buckets, which are similar to directories.

Buckets must have globally unique names. The ec2-upload-bundle utility uploads the bundled AMI to a specified bucket. If the specified bucket does not exist, it will be created. If the specified bucket exists and belongs to another user, the ec2-upload-bundle command will fail.

#### To upload the bundled AMI

• Enter the following command:

```
# ec2-upload-bundle -b <bucket> -m image.manifest.xml -a <access_key> -S <sec
```

where *<bucket>* is the target bucket, *<access\_key>* is your AWS Access Key, and *<secret\_key>* is your AWS Secret Key.

The AMI manifest file and all image parts are uploaded to Amazon S3. The manifest file is encrypted with the Amazon EC2 public key before being uploaded.

# **Sharing AMIs**

#### **Topics**

- Shared AMI Guidelines
- Sharing AMIs
- Making an AMI Public
- Sharing an AMI with Specific Users
- Publishing Shared AMIs

This section describes how to build and share AMIs.

Shared AMIs are AMIs that developers build and make available for other AWS developers to use. Building safe, secure, useable AMIs for public consumption is a fairly straightforward process, if you follow a few simple guidelines.

For information on building shared AMIs, see <u>Shared AMI Guidelines</u>. For information on sharing AMIs, see <u>Sharing AMIs</u>

#### **Shared AMI Guidelines**

These guidelines are not requirements and you are welcome to follow or ignore them. However, following these guidelines should produce a better user experience and help ensure your users' instances are secure.

To build a shared AMI, follow these guidelines:

- 1. Update the AMI Tools at Boot Time
- 2. <u>Disable Password-Based Logins for Root</u>
- 3. Install Public Key Credentials
- 4. Disable sshd DNS Checks (optional)
- 5. Identify Yourself
- 6. Protect Yourself



These guidelines are written for Fedora distributions, but the principles apply to any AMI. You might need to modify the provided examples for other distributions. For other distributions, review their documentation or search the AWS forums in case someone else has done it already.

#### **Update the AMI Tools at Boot Time**

We recommend that your AMIs download and upgrade the Amazon EC2 AMI creation tools during startup. This ensures that new AMIs based on your shared AMIs will have the latest AMI tools.

#### To update the AMI tools at startup on Fedora

Add the following to rc.local:

```
# Update the Amazon EC2 AMI creation tools
echo " + Updating ec2-ami-tools"
wget http://s3.amazonaws.com/ec2-downloads/ec2-ami-tools.noarch.rpm
rpm -Uvh ec2-ami-tools.noarch.rpm && \
echo " + Updated ec2-ami-tools"
```

Use this method to automatically update other software on your image.



When deciding which software to automatically update, consider the amount of WAN traffic that the update will generate (your users will be charged for it) and the risk of the update breaking other software on the AMI.



The above procedure applies to Fedora distributions. For other distributions:

- On most Red Hat systems, add these steps to your /etc/rc.d/rc.local
- On Gentoo systems, add them to /etc/conf.d/local.local.
- On Ubuntu systems, add them to /etc/rc.local.
- On Debian, you might need to create a start up script in /etc/init.d and use update-rc.d <scriptname> defaults 99 (where <scriptname> is the name of the script you created) and add the steps to this script.

#### **Disable Password-Based Logins for Root**

Using a fixed root password for a public AMI is a security risk that can quickly become known. Even relying on users to change the password after the first login opens a small window of opportunity for potential abuse.

To solve this problem, disable password-based logins for the root user. Additionally, we recommend you randomize the root password at boot.

#### To disable password-based logins for root

1. Open the /etc/ssh/sshd\_config file with a text editor and change the following line:

```
#PermitRootLogin yes
```

to

```
PermitRootLogin without-password
```

The location of this configuration file might differ for your distribution, or if you are not running OpenSSH. If this is the case, consult the relevant documentation.

2. To randomize the root password, add the following to your boot process:

```
if [ -f "/root/firstrun" ] ; then
 dd if=/dev/urandom count=50|md5sum|passwd --stdin root
  rm -f /root/firstrun
 echo "* Firstrun *" && touch /root/firstrun
fi
```



This step assumes that a /root/firstboot file is bundled with the image. If file was not created, the root password will never be randomized and will be set to the default.

#### Note

If you are using a distribution other than Fedora, you might need to consult the documentation that accompanied the distribution.

#### **Install Public Key Credentials**

After configuring the AMI to prevent logging in using a password, you must make sure users can log in using another mechanism.

Amazon EC2 allows users to specify a public-private key pair name when launching an instance. When a valid key pair name is provided to the RunInstances API call (or through the command line API tools), the public key (the portion of the key pair that Amazon EC2 retains on the server after a call to CreateKeyPair) is made available to the instance through an HTTP query against the instance metadata.

To login through SSH, your AMI must retrieve the key value at boot and append it to /root/.ssh/authorized\_keys (or the equivalent for any other user account on the AMI). Users will be able to launch instances of your AMI with a key pair and log in without requiring a root password.

This can be applied to any user account; you do not need to restrict it to root.



Rebundling an instance based on this image includes the key with which it was launched. To prevent the key's inclusion, you must clear out (or delete) the authorized\_keys file or exclude this file from rebundling.

#### **Disable sshd DNS Checks (optional)**

Disabling sshd DNS checks slightly weakens your sshd security. However, if DNS resolution fails, SSH logins will still work. If you do not disable sshd checks, DNS resolution failures prevent all logins.

#### To disable sshd DNS checks

• Open the /etc/ssh/sshd\_config file with a text editor and change the following line:

**#UseDNS** yes to: UseDNS no



The location of this configuration file can differ for your distribution or if you are not running OpenSSH. If this is the case, consult the relevant documentation.

#### **Identify Yourself**

Currently, there is no easy way to know who provided a shared AMI as each AMI is represented by a numeric user ID.

We recommend that you post a description of your AMI, and the AMI ID, in the Amazon EC2 developer forum. This provides a convenient central location for users who are interested in trying new shared AMIs.

#### **Protect Yourself**

The previous sections described how to make your shared AMIs safe, secure, and useable for the users who launch them. This section describes guidelines to protect yourself from the users of your AMI.

We recommend against storing sensitive data or software on any AMI that you share. Users who launch a shared AMI might be able to rebundle it and register it as their own. Follow these guidelines to help you to avoid some easily overlooked security risks:

- Always delete the shell history before bundling. If you attempt more than one bundle upload in the same image, the shell history contains your secret access key.
- Bundling a running instance requires your private key and X.509 certificate. Put these and other credentials in a location that is not bundled (such as the ephemeral store).
- Exclude the ssh authorized keys when bundling the image. The Amazon public images store the public key used to launch an instance with its ssh authorized keys file.



Unfortunately, it is not possible for this list of guidelines to be exhaustive. Build your shared AMIs carefully and take time to consider where you might expose sensitive data.

# **Sharing AMIs**

Amazon EC2 enables users to share their AMIs with other users. This section describes how to share AMIs using the Amazon EC2 command line tools.



Before proceeding, make sure to read the security considerations of sharing AMIs in the **Shared AMI Guidelines** section.

AMIs have a launchPermission property that controls which users, besides the owner, are allowed to launch instances of that AMI. By modifying an AMI's launchPermission property, you can allow all users to launch the AMI (make the AMI public) or only allow a few specific users to launch the AMI.

The launchPermission attribute is a list of users and launch groups. Launch permissions can be granted by adding or removing items from the list. Explicit launch permissions for users are granted or revoked by adding or removing their AWS account IDs. The only launch group currently supported is the all group, which makes the AMI public. The rest of this section refers to launch groups simply as groups. Launch groups are not the same as security groups and the two should not be confused. An AMI can have both public and explicit launch permissions.



You are not billed when your AMI is launched by other users. Users launching the AMI are billed.

#### Select from the following:

- 1. Making an AMI Public
- 2. Sharing an AMI with Specific Users
- 3. Publishing Shared AMIs

## Making an AMI Public

#### To make an AMI public

• Add the all group to the AMI's launchPermission attribute using the <u>ec2-modify-image-attribute</u> command:

```
PROMPT> ec2-modify-image-attribute <ami_id> --launch-permission -a all
```

where <ami\_id> is the ID of the AMI.

#### To check the launch permissions of an AMI

• Use the <u>ec2-describe-image-attribute</u> command:

```
PROMPT> ec2-describe-image-attribute <ami_id> -l
```

where <ami\_id> is the ID of the AMI.

#### To make an AMI private again

• Remove the all group from its launch permissions:

```
PROMPT> ec2-modify-image-attribute <ami_id> -l -r all
```

where <ami\_id> is the ID of the AMI.

This will not affect any explicit launch permissions for the AMI or any running instances of the AMI.

#### **Example**

This example makes the ami-2bb65342 AMI public:

```
PROMPT> ec2-modify-image-attribute ami-2bb65342 --launch-permission -a all launchPermission ami-2bb65342 ADD group all
```

This examples displays the launch permissions of the ami-2bb65342 AMI:

```
PROMPT> ec2-describe-image-attribute ami-2bb65342 -l launchPermission ami-2bb65342 group all
```

This example removes the all group from the permissions of the ami-2bb65342 AMI, making it private:

```
PROMPT> ec2-modify-image-attribute ami-2bb65342 -l -r all launchPermission ami-2bb65342 REMOVE group all
```

# **Sharing an AMI with Specific Users**

You can share an AMI with specific users without making the AMI public. All you need is the user's AWS user's account ID, which is available on the AWS Account Activity page.

#### To grant explicit launch permissions

• Enter the following command:

```
PROMPT> ec2-modify-image-attribute <ami_id> -l -a <user_id>
```

where <ami\_id> is the ID of the AMI and <user\_id> is the user's account ID, without hyphens.

#### To remove launch permissions for a user

• Enter the following command:

```
PROMPT> ec2-modify-image-attribute <ami_id> -l -r <user_id>
```

where <ami\_id> is the ID of the AMI and <user\_id> is the user's account ID, without hyphens.

#### To remove all launch permissions

• Enter the following command to remove all public and explicit launch permissions:

```
PROMPT> ec2-reset-image-attribute <ami_id> -l
```

where <ami\_id> is the ID of the AMI.



The AMI owner always has rights to the AMI and will be unaffected by the

#### **Example**

The following example grants launch permissions to the 495219933132 user for the ami-2bb65342 AMI:

```
PROMPT> ec2-modify-image-attribute ami-2bb65342 -l -a 495219933132 launchPermission ami-2bb65342 ADD userId 495219933132
```

The following example removes launch permissions from the 495219933132 user for the ami-2bb65342 AMI:

```
PROMPT> ec2-modify-image-attribute ami-2bb65342 -l -r 495219933132 launchPermission ami-2bb65342 REMOVE userId 495219933132
```

The following example removes all public and explicit launch permissions from the ami-2bb65342 AMI:

```
PROMPT> ec2-reset-image-attribute ami-2bb65342 -l launchPermission ami-2bb65342 RESET
```

## **Publishing Shared AMIs**

After creating a shared AMI, other developers can find it in the <u>EC2 Resource</u> Center.

To publish your AMI, post it in the Public AMIs Folder of the <u>Amazon Web</u> Services Resource Center.

You must include the following information when publishing AMIs:

- AMI ID
- AMI manifest

We recommend including the following information when publishing AMIs:

- Publisher
- Publisher URL
- OS / Distribution
- Key Features
- Description
- Daemons / Services
- Release Notes

You can cut and paste the following template into the document. You must be in HTML edit mode.

```
| Compared to the state of 
                                                     This image contains the following daemons / services:
                                                       ul>
                                                                                                              | Compared to the compared
                                                                                                              | Compared to the compared
                                                       <h2><strong>What&#39;s New?</strong></h2>The following changes were made
ul>
                                                       |Release Notes 1|
<span style="font-size: x-small; font-family: courier new,courier">&nbs;
<span style="font-size: x-small; font-family: courier new,courier">&nbs;
<span style="font-size: x-small; font-family: courier new,courier">&nbs;
ul>
```

# **Creating Paid AMIs**

#### Introduction

A *paid AMI* is an AMI that you sell to other Amazon EC2 users. They pay you according to the price you set. To be able to create a paid AMI, you use Amazon DevPay. What is Amazon DevPay?

Amazon DevPay is a billing and account management service that enables you to get paid for an AMI you create and that other Amazon EC2 users use. Amazon DevPay creates and manages the order pipeline and billing system for you. Your customers sign up for your AMI, and Amazon DevPay automatically meters their usage of Amazon EC2, bills them based on the pricing you set, and collects their payments. What else is special about DevPay?

- You can charge customers for your AMI; the charges can include recurring charges based on the customer's usage Amazon EC2, a fixed one-time charge, and a recurring monthly charge.
- Your customers can easily sign up and pay for your AMI with their trusted Amazon.com accounts.
- Your customers are authenticated, thus ensuring they have access only to what they should.
- If your customers don't pay their bills, DevPay turns off their access to your AMI for you.
- Amazon Payments handles payment processing.



#### **Basic DevPay Flow**

	indicates that you have teamed up with Amazon Payments to make billing easy and secure.		
2	Your customer pays the price you've defined to use your product.		
3	DevPay subtracts the required DevPay fees and pays you the difference.		
4	You pay the costs of Amazon EC2 that your AMI consumed.		

To learn more about Amazon DevPay, see the *Amazon DevPay Developer Guide*.

## **Paid and Supported AMIs**

Amazon DevPay and Amazon EC2 work together to support these two scenarios:

- **Paid AMIs**—You charge for the use of an AMI you've created and shared with either select customers or the public. You might want to share the AMI with only select customers, for example, if you're offering a special price to just those customers.
- **Supported AMIs**—You charge your customers for software or a service you provide that they use with their own AMIs

You determine the rates you want to charge customers who use paid AMIs or supported AMIs. The price you charge can include a one-time charge, a monthly charge, and a markup on the hourly instance or the data transferred charge. For more information about how you can set your prices, see the *Amazon DevPay Developer Guide*.

# **Summary of How Paid AMIs Work**

The following steps summarize the basic flow for creating and using paid AMIs:



Detailed information about most of the following steps is provided in the Amazon DevPay Developer Guide.

#### **Paid AMI Process**

- You create an AMI as described elsewhere in this guide.
- You register a product with Amazon DevPay. As part of this process, you provide a product description, product pricing, etc. This registration process creates a product code for the product and a URL where customers can sign up to use the product (called the *purchase URL*).
- You use an Amazon EC2 command or API call to associate the product code with your AMI. This makes the AMI a paid AMI.
- You use an Amazon EC2 command or API call to share the AMI with select customers or the public.



Even if you share a paid AMI and it has a product code, no one can use the AMI until they sign up for it (see the following steps).

- You advertise your paid AMI to customers and make the purchase URL available to them. Note that you can submit your paid AMI to be listed on the AWS Resource Center with other public AMIs at http://developer.amazonwebservices.com/connect/kbcategory.jspa?categoryID=116.
- Customers who want to use your AMI discover your product through your advertisements or the AWS Resource Center, etc.
- Customers then use the purchase URL you provide to sign up for and purchase your product. If they're not already signed up for Amazon EC2, they'll be prompted to sign up. They purchase your product with their Amazon.com accounts. They must have the credentials needed to launch Amazon EC2 instances. At this point, they have the AMI ID (from either step 5 or step 6 above).
- Customers then launch an Amazon EC2 instance specifying the AMI ID. Because you associated the shared AMI with the product code, the customers are charged at the rate you set. For more information, see Paying for AMIs

Each customer's bill for the AMI is displayed on their Application Billing page,

which shows the activity for DevPay products. For more information, see the *Amazon DevPay Developer Guide*.

# Summary of How Supported AMIs Work

The following steps summarize the basic process for supported AMIs:

#### **Supported AMI Process**

- You <u>register a product</u> with Amazon DevPay. As part of this process, you provide a product description, product pricing, etc. This registration process creates a product code for the product and a URL where customers can sign up to use the product (called the *purchase URL*).
- 2 You then advertise your paid AMI to customers and make the purchase URL available to them.
- 3 Customers who want to use your product discover the product through your advertisements, etc.
- 4 Customer then use the purchase URL to sign up for and purchase your product. If they're not already signed up for Amazon EC2, they'll be prompted to sign up. They purchase your product with their Amazon.com accounts. They must have the credentials needed to launch Amazon EC2 instances. At this point, they have the product code (from either step 2 or step 3 above).
- 5 Customers than use an Amazon EC2 command or API call to <u>associate the product code</u> with their AMIs.
- 6 Customers then launch one or more instances of the AMIs. Because the customers associated their AMIs with the product code, they are charged at the rate you set.

Each customer's bill for the AMI is displayed on their Application Billing page, which shows the activity for DevPay products. For more information, see the *Amazon DevPay Developer Guide*.

When a customer contacts you for support for an AMI, you can confirm your product code is associated with the AMI and the customer's instance is currently running the AMI (see <u>Confirming an Instance Is Running an AMI Associated</u> with a <u>Product Code</u>).

# **Product Registration**

You must register your AMI (referred to as your product) with Amazon DevPay. During registration, you provide product information such as pricing, and you receive information you need to sell your product.



AWS must approve your product after you register it. The approval process takes several business days. During that time you can begin integrating your product with DevPay.

You provide the following information during registration:

- Company name
- Product name
- Product description (as you want your customers to see it)
- Redirect URL (the page you want customers to see after they have purchased the product)
- Any terms and conditions you want displayed (optional)
- Contact e-mail address and telephone number (to be used by AWS and not displayed to customers)
- Contact e-mail or URL (to be displayed to customers)
- Pricing for use of the product

The information you display at the redirect URL should give information about the AMI.

Registration provides you with the following information:

- Product code
- Product token

#### • Purchase URL

You need the product code and purchase URL to integrate your product with DevPay as described in **Summary of How Paid AMIs Work** and **Summary of** How Supported AMIs Work. You need the product token if you're going to set up your system to later verify whether a customer is still subscribed to your product. For more information, see the *Amazon DevPay Developer Guide*.

#### (I) Important

The Amazon DevPay Developer Guide covers the procedure for registering your product with Amazon DevPay. Before you register your product, we recommend you read the information in that guide about how to set your AMI's price and how billing for Amazon DevPay products works.

# Associating a Product Code with an AMI

You must be the owner of an AMI to associate a product code with it. You can associate a single product code with more than one paid AMI. You might do this if you have similar versions of an AMI (for example, a 32-bit version and a 64-bit version), you've assigned them all the same price, and you'd like to minimize the number of Amazon DevPay product codes you have (to make your bookkeeping easier).

#### To associate a product code with an AMI

• Enter the following command:

```
PROMPT> ec2-modify-image-attribute <ami_id> --product-code product_code>
```

where <ami\_id> is the AMI ID and duct\_code> is the product code.

#### To verify the product code is associated with the AMI

• Enter the following command:

```
PROMPT> ec2-describe-image-attribute <ami_id> --product-code
```

You can't change or remove the productCodes attribute after you've set it. If you want to use the same image without the product code or associate a different product code with the image, you must reregister the image to obtain a new AMI ID. You can then use that AMI without a product code or associate the new product code with the AMI ID.

#### Example

The following example associates the ami-5bae4b32 AMI with the 774F4FF8 product code:

```
PROMPT> ec2-modify-image-attribute ami-5bae4b32 --product-code 774F4FF8

productCodes ami-5bae4b32 productCode 774F4FF8
```

This example verifies that the product code is associated with the AMI:

```
PROMPT> ec2-describe-image-attribute ami-5bae4b32 --product-code
productCodes ami-5bae4b32 productCode 774F4FF8
```

# Sharing Your Paid AMI with Select Users or the Public

After you associate the product code with the AMI, you need to share the AMI with select customers or the public by using the ec2-modify-image-attribute command.

#### To share the AMI

• Enter the following command:

```
PROMPT> ec2-modify-image-attribute <ami_id> --launch-permission -a all
```

where <ami id> is the AMI ID.

Even though you've shared the AMI, no one can use it until they sign up for your product by going to the purchase URL. Once customers sign up, any instances of the paid AMI they launch will be billed at the rate you specified during product registration.

#### **Example**

The following example shares the ami-5bae4b32 AMI with the public:

```
PROMPT> ec2-modify-image-attribute ami-5bae4b32 --launch-permission -a all launchPermission ami-5bae4b32 ADD group all
```

# Confirming an Instance Is Running an AMI Associated with a Product Code

If you have created a product for others to use with their AMIs (the supported AMI scenario), you might want to confirm that a particular AMI is associated with your product code and a particular instance is currently running that AMI.



You must be the owner of the product code to successfully call **ec2-confirm-product-instance** with that product code.

Because your customers don't own the product code, they should describe their instances to confirm their instances are running with your product code.

# To confirm an AMI is associated with your product code and an instance is running that AMI

• Enter the following command:

where code is the product code and <instance</pre> is the instance.

If the AMI is associated with the product code, true is returned with the AMI owner's account ID. Otherwise, false is returned.

#### Example

The following example confirms whether the i-10a64379 instance is running the 6883959E product code:

```
PROMPT> ec2-confirm-product-instance 6883959E -i i-10a64379
6883959E i-10a64379 true 495219933132
```

# **Getting the Product Code from Within an Instance**

A running Amazon EC2 instance can determine if an Amazon DevPay product code is attached to itself. The instance retrieves the product code like it retrieves its other metadata. For more information about the retrieval of metadata, see Instance Metadata.

The instance retrieves the product code by querying a web server with this REST-like API call:

GET http://169.254.169.254/2007-03-01/meta-data/product-codes

Example response:

774F4FF8

# **Launching and Using Instances**

#### **Topics**

- Instance Usage
- Instance Types
- Instance Metadata
- <u>Using Shared AMIs</u>
- Paying for AMIs
- Get Console Output and Reboot Instances

This section describes how to launch instances and retrieve instance-specific data from within the instance. It also covers launching shared AMIs and security risks associated with running shared AMIs.

#### **Instance Usage**

The instance is your basic computation building block. Amazon EC2 offers multiple instance types from which you can choose. You can run as many or as few instances as you need at any given time.

For information about available instance types, see <u>Selecting Instance Types</u>.

Once launched, an instance looks very much like a traditional host. You have complete control of your instances; you have root access to each one and you can interact with them as you would any machine.

Here are some suggestions for making the best use of Amazon EC2 instances.

• Do not rely on an instance's local storage for valuable, long-term data.

When instances fail, the data on the local disk is lost. Use a replication strategy across multiple instances to keep your data safe or store your persistent data in Amazon S3

• Define images based on the type of work they perform.

For "internet applications," you might define one image for database instances and another for web servers. Image creation and storage are cheap and easy operations, so you can individualize and customize as necessary. Specialized images can result in smaller AMI sizes, which will boot considerably faster.

• Monitor the health of your instances.

You can make your instances work for you by configuring them to monitor each other. For example, you could create an image that contains an open-source monitoring tool such as Nagios or OpenNMS. Then, your other instances could report their health to the monitoring instance.

• Keep your Amazon EC2 firewall permissions as restrictive as possible.

Only open up permissions that you require. Use separate groups to deal with instances that have different security requirements. Consider using

additional security measures inside your instance (such as using your own firewall). If you need to log in interactively (ssh), consider creating a bastion security group that allows external login and keep the remainder of your instances in a group that does not allow external login.

# **Instance Types**

Amazon EC2 provides the flexibility to meet your computing needs by allowing you to choose from different instance types. Each instance type provides a predictable amount of dedicated compute capacity and is charged per instance-hour consumed.

#### **Available Instance Types**

The following instance types are available:

	Small	Large	Extra Large
CPU	1 EC2 Compute Unit (1 virtual core with 1 EC2 Compute Unit)	4 EC2 Compute Units (2 virtual cores with 2 EC2 Compute Units each)	8 EC2 Compute Units (4 virtual cores with 2 EC2 Compute Units each)h
Memory	1.7 GB	7.5 GB	15 GB
Storage	160 GB instance storage (150 GB plus 10 GB root partition)	850 GB instance storage (2 x 420 GB plus 10 GB root partition)	1690 GB instance storage (4 x 420 GB plus 10 GB root partition)
Platform	32-bit	64-bit	64-bit
I/O Performance	Moderate	High	High
Instance Type Name	m1.small	m1.large	m1.xlarge



The small instance type is the original Amazon EC2 instance type available since the launch of Amazon EC2. It is the default instance type for all customers. To use other instance types, you must specify them through the RunInstances operation.

#### **Measuring Compute Resources**

Transitioning to a utility computing model changes how developers are trained to think about CPU resources. Instead of purchasing or leasing a particular processor to use for several months or years, you are renting capacity by the hour. Because Amazon EC2 is built on commodity hardware, over time there might be several different types of physical processors underlying different virtual EC2 instances. Our goal is to provide a consistent amount of CPU capacity regardless of the actual underlying hardware.

Amazon EC2 uses a variety of measures to provide each instance with a consistent and predictable amount of CPU capacity. To make it easy for developers to compare CPU capacity between different instance types, we defined an Amazon EC2 Compute Unit.



We use several internal benchmarks and tests to manage the consistency and predictability of the performance of an Amazon EC2 Compute Unit. For more information, refer to the <a href="Instance page">Instance page</a>.

To find out which instance will work best for your application, we recommend launching an instance and using your own benchmark application. This will help you determine which instance type works best for your specific use case.

#### I/O Resources

Amazon EC2 provides virtualized server instances. While some resources like CPU, memory and instance storage are dedicated to a particular instance, other resources like the network and the disk subsystem are shared amongst instances. If each instance on a physical host tries to use as much of one of these shared resources as possible, each will receive an equal share of that resource. However, when a resource is under-utilized you will often be able to consume a higher share of that resource while it is available.

The different instance types will provide higher or lower minimum performance from the shared resources depending on their size. Each of the instance types has an I/O performance indicator (moderate or high). Instance types with high I/O performance have a larger allocation of shared resources. Allocating larger share of shared resources also reduces the variance of I/O performance. For most applications, moderate I/O performance is more than enough. However, for applications that require greater or more consistent I/O performance, consider instances with high I/O performance.

# **Instance Metadata**

Amazon EC2 instances can access instance-specific metadata as well as data supplied when launching the instances. This data can be used to build more generic AMIs that can be modified by configuration files supplied at launch time.

#### **Example**

If you run web servers for various small businesses, they can all use the same AMI and retrieve their content from the Amazon S3 bucket you specify at launch.

To add a new customer at any time, simply create a bucket for the customer, add their content, and launch your AMI.

# **Categories of Available Data**

The data available to instances is categorized into metadata and user-supplied data.

Metadata is specific to an instance. Currently, we provide:

Data	Description	Version Introduced
ami-id	The AMI ID used to launch the instance.	1.0
ami- manifest- path	The manifest path of the AMI with which the instance was launched.	1.0
ami- launch- index	The index of this instance in the reservation (per AMI).	1.0
ancestor- ami-ids	The AMI IDs of any instances that were rebundled to create this AMI.	2007-10-10
instance-id	The ID of this instance.	1.0
instance- type	The type of instance to launch. For more information, see <u>Selecting Instance Types</u> .	2007-08-29
local- hostname	The local hostname of the instance.	2007-01-19
public- hostname	The public hostname of the instance.	2007-01-19
local-ipv4	Public IP address if launched with direct addressing; private IP address if launched with public addressing.	1.0
public-ipv4	NATted public IP Address	2007-01-19
public- keys/	Public keys. Only available if supplied at instance launch time	1.0
reservation- id	ID of the reservation.	1.0
security- groups	Names of the security groups the instance is launched in. Only available if supplied at instance launch time	1.0
product- codes	Product codes associated with this instance.	2007-03-01

User-supplied data is treated as opaque data: what you give us is what you get back.

#### Note

- All instances launched together get the same user-supplied data. You can use the AMI launch index as an index into the data.
- User data is limited to 16K. This limit applies to the data in raw form, not base64 encoded form.
- The user data must be base64 encoded before being submitted to the API. The API command-line tools perform the base64 encoding for you. The data is base64 decoded before presented to the instance.

#### **Data Retrieval**

An instance retrieves the data by querying a web server using a Query API. The base URI of all requests is http://169.254.169.254/2007-08-29/ where 2007-08-29 indicates the API version.



Amazon EC2 Version 1.0 is part of a legacy versioning scheme. Newer versions follow a date based versioning scheme. For more information on the versioning scheme used by Amazon EC2, see <u>API Versioning</u>.

The latest version of the API is always available using the URI http://169.254.169.254/latest.

#### **Security of Launch Data**

Although only your specific instance can access launch data, the data is not protected by cryptographic methods. You should take suitable precautions to protect sensitive data (such as long lived encryption keys).



You are not billed for HTTP requests used to retrieve metadata and usersupplied data.

#### **Retrieving Metadata**

Requests for a specific metadata resource returns the appropriate value or a 404 HTTP error code if the resource is not available. All metadata is returned as text (content type text/plain).

Requests for a general metadata resource (i.e. an URI ending with a /) return a list of available resources or a 404 HTTP error code if there is no such resource. The list items are on separate lines terminated by line feeds (ASCII 10).

#### Example

The following examples list HTTP GET requests and responses. You can use a

tool such as curl or wget to make these types of requests.

This example gets the available API versions:

```
GET http://169.254.169.254/
1.0
2007-01-19
2007-03-01
2007-08-29
```

This example gets the top-level metadata items:

```
GET http://169.254.169.254/2007-08-29/meta-data/
ami-id
ami-launch-index
ami-manifest-path
hostname
instance-id
instance-type
local-hostname
local-ipv4
public-hostname
public-ipv4
public-keys/
reservation-id
security-groups
```

This examples get the value of each metadata item in the above example:

```
GET http://169.254.169.254/2007-08-29/meta-data/ami-manifest-path my-amis/spamd-image.manifest.xml

GET http://169.254.169.254/2007-08-29/meta-data/ami-manifest-path my-amis/spamd-image.manifest.xml

GET http://169.254.169.254/2007-08-29/meta-data/ami-id ami-2bb65342

GET http://169.254.169.254/2007-08-29/meta-data/reservation-id r-fea54097

GET http://169.254.169.254/2007-08-29/meta-data/hostname domU-12-34-31-00-00-05.usmal.compute.amazonaws.com
```

This example gets the list of available public keys:

```
GET http://169.254.169.254/2007-08-29/meta-data/public-keys/
0=my-public-key
```

This example shows the formats in which public key 0 is available:

```
GET http://169.254.169.254/2007-08-29/meta-data/public-keys/0/
openssh-key
```

This example gets public key 0 (in openssh-key format):

```
GET http://169.254.169.254/2007-08-29/meta-data/public-keys/0/openssh-key
ssh-rsa AAAA....wZEf my-public-key
```

This example gets the product code:

```
GET http://169.254.169.254/2007-08-29/meta-data/product-codes
774F4FF8
```

#### **Retrieving User Data**

Requests for the user data returns the data as-is (content type application/xoctetstream).



All user-supplied data is treated as opaque data; what you give us is what you get back. It is the responsibility of the instance to interpret this data appropriately.

#### **Example**

This shows an example of returning comma-separated user-supplied data:

```
GET http://169.254.169.254/2007-08-29/user-data
1234, fred, reboot, true | 4512, jimbo, | 173, , ,
```

This shows an example of returning line-separated user-supplied data:

```
GET http://169.254.169.254/2007-08-29/user-data
[general]
instances: 4

[instance-0]
s3-bucket: fred

[instance-1]
reboot-on-error: yes
```

### **Example: AMI Launch Index Value**

In this example, Alice wants to launch four instances of her favorite database AMI with the first acting as master and the remainder acting as replicas.

The master database configuration specifies various database parameters (e.g., the size of store) while the replicas' configuration specifies different parameters, such as the replication strategy. Alice decides to provide this data as an ASCII string with a pipe symbol (| delimiting the data for the various instances:

```
store-size=123PB backup-every=5min | replicate-every=1min | replicat
```

where store-size=123PB backup-every=5min defines the master database configuration, replicate-every=1min defines the first replicant's configuration, replicate-every=2min defines the second replicant's configuration, and so on.

Alice launches four instances:

```
$ ec2-run-instances ami-2bb65342 -n 4 -d "store-size=123PB backup-every=
RESERVATION r-fea54097 598916040194 default
INSTANCE i-3ea74257 ami-2bb65342 pending 0 ml.small 2007-08-07T11:29:58-
INSTANCE i-31a74258 ami-2bb65342 pending 1 ml.small 2007-08-07T11:29:58-
INSTANCE i-31a74259 ami-2bb65342 pending 2 ml.small 2007-08-07T11:29:58-
INSTANCE i-31a7425a ami-2bb65342 pending 3 ml.small 2007-08-07T11:29:58-
```

Once launched, all instances have a copy of the user data and the common metadata shown here:

• AMI id: ami-2bb65342

• AMI manifest path: ec2-public-images/getting-started.manifest.xml

• Reservation ID: r-fea54097

Public keys: none

• Security group names: default

• Instance type: m1.small

#### However each instance has certain unique metadata:

#### Instance 1

Metadata	Value
instance-id	i-3ea74257
ami-launch-index	0
hostname	domU-12-43-33-00-01-27.usma1.compute.amazonaws.com
local-ipv4	216.182.228.87

#### Instance 2

Metadata	Value
instance-id	i-31a74258
ami-launch-index	1
hostname	domU-12-31-33-00-01-72.usma1.compute.amazonaws.com
local-ipv4	216.182.228.88

#### Instance 3

Metadata	Value
instance-id	i-31a74259
ami-launch-index	2
hostname	domU-12-31-33-00-01-73.usma1.compute.amazonaws.com
local-ipv4	216.182.228.89

#### Instance 4

Metadata	Value
instance-id	i-31a7425a

ami-launch-index	3
hostname	domU-12-31-33-00-01-74.usma1.compute.amazonaws.com
local-ipv4	216.182.228.90

Therefore, an instance can determine its portion of the user-supplied data by doing the following:

#### **Metadata Discovery Process**

```
Determine which instance in the launch group it is:

GET http://169.254.169.254/2007-08-29/meta-data/ami-launch-index

Retrieve the user data:

GET http://169.254.169.254/2007-08-29/user-data store-size=123PB backup-every=5min | replicate-every=1min

Extract the appropriate part of the user data:

user_data.split('|')[ami_launch_index]
```

# **Instance Storage**

Every instance includes a fixed amount of storage space on which you can store data. Within this document, it is referred to as the "ephemeral store" as it is not designed to be a permanent storage solution.

If an instance reboots (intentionally or unintentionally), the data on the ephemeral store will survive. If the underlying drive fails or the instance is terminated, the data will be lost.

We highly recommend backing up important data to Amazon S3.

# **Storage Locations**

Storage is exposed on the following instance types as follows:

Location	Description
/dev/sda1	Formatted and mounted as root (/) on all instance types
/dev/sda2	Formatted and mounted as /mnt on small instances
/dev/sda3	Formatted and mounted as /swap on small instances
/dev/sdb	Formatted and mounted as /mnt on large and extra large instances
/dev/sdc	Available on large and extra large instances; not mounted
/dev/sdd	Available on extra large instances; not mounted
/dev/sde	Available on extra large instances; not mounted

#### **Disk Performance Optimization**

Due to how Amazon EC2 virtualizes disks, the first write to any location on an instance's drives will perform slower than subsequent writes. For most applications, amortizing this cost over the lifetime of the instance will be acceptable. However, if you require high disk performance, we recommend initializing drives by writing once to every drive location before production use.

To initialize the stores, use the following commands on the m1.large and m1.xlarge instance types:

To perform the initialization on all drives at the same time:

```
dd if=/dev/zero bs=1M|tee /dev/sdb|tee /dev/sdc|tee /dev/sde > /dev/sdd
```



This can take a long time (about 8 hours for an extra-large instance).

# **RAID Configuration**

Configuring drives for RAID initializes them by writing to every drive location. When configuring software-based RAID, make sure to change the minimum reconstruction speed:

echo \$((30\*1024)) > /proc/sys/dev/raid/speed\_limit\_min



You cannot use iostat (part of the sar System Activity Reporting package) to watch performance. You also cannot watch 'cat /proc/mdstat'.

# **Using Shared AMIs**

This section describes how to find and safely use shared AMIs. One of the easiest ways to get started with Amazon EC2 is to use a shared AMI that has the components you need and add custom content.

### **Finding Shared AMIs**

#### To find shared AMIs

• Use the **ec2-describe-images** command (or the abbreviated **ec2dim** command) with a flag to filter the results.

#### **Example**

This command displays a list of all public AMIs.

```
PROMPT> ec2dim -x all
```

The -*x all* flag shows AMIs executable by all users. This includes AMIs you own.

This command displays a list of AMIs for which you have explicit launch permissions:

```
PROMPT> ec2dim -x self
```

AMIs that you own are excluded from the list.

This command displays a list of AMIs owned by Amazon:

```
PROMPT> ec2dim -o amazon
```

This command displays a list of AMIs owned by a particular user:

```
PROMPT> ec2dim -0 <target_uid>
```

where <target\_uid> is the account ID of the user who owns the AMIs for which you are looking.

#### Safe Use of Shared AMIs

AMIs are launched at the user's own risk. Amazon cannot vouch for the integrity or security of AMIs shared by other users. Therefore, you should treat shared AMIs as you would any foreign code that you might consider deploying in your own data center and perform the appropriate due diligence.

Ideally, you should get the AMI ID from a trusted source (a website, another user, etc). If you do not know the source of an AMI, we recommended that you search the forums for comments on the AMI before launching it. Conversely, if you have questions or observations about a shared AMI, feel free to use the AWS forums to ask or comment.

Amazon's public images have an aliased owner and display amazon in the userId field. This allows you to find Amazon's public images easily.



Users cannot alias an AMI's owner.

If you choose to launch a shared AMI, take the following minimum steps after launch to confirm the AMI is not doing anything malicious:

- 1. Check the ssh authorized keys file. The only key in the file should be the key you used to launch the AMI.
- 2. Check open ports and running services.
- 3. Change the root password if is not randomized on startup. For more information on randomizing the root password on startup, see <u>Disable Password-Based Logins for Root</u>.
- 4. Check if ssh allows root password logins. See <u>Disable Password-Based</u> <u>Logins for Root</u> for more information on disabling root based password logins.
- 5. Check whether there are any other user accounts that might allow backdoor entry to your instance. Accounts with super user privileges are particularly

dangerous.

6. Verify that all cron jobs are legitimate.

# **Paying for AMIs**

- Finding Paid AMIs
- Purchasing a Paid AMI
- Launching Paid AMIs
- Paying for Support
- Bills for Paid and Supported AMIs

Amazon EC2 integrates with Amazon DevPay, allowing developers to charge users for the use of their AMIs or to provide support for instances. To learn more about Amazon DevPay see the *Amazon DevPay Developer Guide*. To learn more about charging for your use of your AMIs, or providing support, see <a href="Creating Paid AMIs">Creating Paid AMIs</a>

This section describes how to discover paid AMIs, launch paid AMIs, and launch instances with a support product code. Paid AMIs are AMIs you can purchase from other developers.

#### **Finding Paid AMIs**

There are several ways you can determine what paid AMIs are available for you to purchase. You can look for information about them on the Amazon EC2 resource center and forums. Alternatively, a developer might give you information about a paid AMI directly.

You can also tell if an AMI is a paid AMI by describing the image with the **ec2-describe-images** command. This command lists the product code associated with an AMI (see the example below). If the AMI is a paid AMI, it has a product code. Otherwise, it does not. You can then go to the Amazon EC2 resource center and forums, which might have more information about the paid Amazon EC2 and where you can sign up to use it.



You must sign up for a paid AMI before you can launch it.

#### To check if an AMI is paid:

• Enter this command:

```
$ ec2-describe-images <ami_id>
```

where <ami\_id> is the AMI ID.

The command returns the following:

where <ami\_id> is the AMI ID, <manifest> is the manifest location, <user\_id> is the ID of the user that owns the AMI, <status> indicates whether the AMI is available, and product\_code is the product code associated with the AMI. If a product code is present, the AMI is a paid AMI.

#### Example

This example shows an **ec2-describe-images** call describing a paid AMI. The product code is 774F4FF8.

\$ ec2-describe-images ami-2bb65342
IMAGE ami-2bb65342 awesome-ami/webserver.manifest.xml 495219933132 available private 774F2

# **Purchasing a Paid AMI**

You must sign up for (purchase) the paid AMI before you can launch it.

Typically a seller of a paid AMI presents you with information about the AMI, its price, and a link where you can buy it. When you click the link, you're first asked to log in with an Amazon.com login, and then you're taken to a page where you see the paid AMI's price and you confirm you want to purchase the AMI.

### **Launching Paid AMIs**

This section describes how to launch paid AMIs and launch instances with a support product code.

After you purchase a paid AMI, you can launch instances of it. Launching a paid AMI is the same as launching any other AMI. No additional parameters are required. The instance will be charged according to the rates set by the owner of the AMI (which will be more than the base Amazon EC2 rate).

#### To launch a paid AMI

• Enter the following command:

```
$ ec2-run-instances <ami_id>
```

where <ami id> is the AMI ID.



The owner of a paid AMI will be able to confirm if a particular instance was launched using their paid AMI.

#### **Example**

This example shows the command used to launch the ami-2bb65342 AMI:

```
$ ec2-run-instances ami-2bb65342
INSTANCE i-10a64379 ami-2bb65342 pending
```

### **Paying for Support**

The paid AMI feature also allows developers to offer support for software (or derived AMIs). Developers can create support products that you can sign up to use. With this model, the developer provides you with a product. During sign-up for the product, the developer gives you a product code for that product, which you must then associate with your own AMI. This allows the developer to confirm that your instance is eligible for support. It also ensures that when you run instances of the product, you are charged according to the developer's terms for the product.

#### To associate the product code with your AMI

• Use the **ec2-modify-image-attribute** command:

```
PROMPT> ec2-modify-image-attribute <ami_id> --product-code code
```

where <ami\_id> is the AMI ID and code is the product code.



Once set, the product code attribute cannot be changed or removed.

To launch a paid AMI, no additional parameters are required for the runinstances. The instance is charged according to the rates set by the AMI owner.

#### Example

The following command associates the *ami-2bb65342* AMI with the *774F4FF8* product code:

```
PROMPT> ec2-modify-image-attribute ami-2bb65342 --product-code 774F4FF8 productCodes ami-2bb65342 productCode 774F4FF8
```

The following command launches the ami-2bb65342 paid AMI:

# \$ ec2-run-instances ami-2bb65342 INSTANCE i-10a64379 ami-2bb65342 pending

# **Bills for Paid and Supported AMIs**

At the end of each month, you receive an e-mail with the amount your credit card has been charged for using the paid or supported AMIs during the month. This bill is separate from your regular Amazon EC2 bill.

At any time you can view the usage information for your paid and supported AMIs at <a href="http://www.amazon.com/dp-applications">http://www.amazon.com/dp-applications</a>.

# Get Console Output and Reboot Instances

Console output is a valuable tool for problem diagnosis. It is especially useful for troubleshooting kernel problems and service configuration issues that could cause an instance to terminate or become unreachable before its SSH daemon can be started.

Similarly, the ability to reboot instances that are otherwise unreachable is valuable for both troubleshooting and general instance management.

Amazon EC2 instances do not have a physical monitor through which you can view their console output. They also lack physical controls that allow you to power up, reboot, or shut them down. To allow these actions, we provide them through the Amazon EC2 SOAP API, Query API, and command-line tools.

### **Get Console Output**

The Amazon EC2 instance console output displays the exact console output that would normally be displayed on a physical monitor attached to a machine. This output is buffered because the instance produces it and then posts it to a store where the instances owner can retrieve it.

The posted output is not continuously updated; only when it is likely to be of the most value. This includes shortly after instance boot, after reboot, and when the instance terminates.



Only the most recent 64 KB of posted output is stored, which is available for at least 1 hour after the last posting.

You can retrieve the console output for an instance through the SOAP API call described in GetConsoleOutput, the Query API call described in GetConsoleOutput, and the command line tool described in ec2-get-consoleoutput.



Only the instance owner can access the console output.

## **Reboot Instances**

Just as you can reset a machine by pressing the reset button, you can reset Amazon EC2 instances through the SOAP API described in RebootInstances, the Query API described in RebootInstances, and the command line tool described in ec2-reboot-instances.

# **Instance Addressing and Network Security**

#### **Topics**

- Instance Addressing
- Network Security

This section provides information on the IP addresses that are assigned to instances and how to configure the firewall provided by Amazon EC2.

# **Instance Addressing**

All Amazon EC2 instances are assigned two IP addresses at launch: a private address (RFC 1918), and a public address. The public IP address is directly mapped to the private address through Network Address Translation (NAT). Private addresses are only reachable from within the Amazon EC2 network. Public addresses are reachable from the Internet.

Amazon EC2 also provides an internal DNS name and a public DNS name which map to the private and public IP addresses respectively. The internal DNS name can only be resolved within Amazon EC2. The public DNS name resolves to the public IP address outside the Amazon EC2 network and the private IP address within the Amazon EC2 network.

For more information, see **Instance Addressing**.



During earlier stages of the Amazon EC2 Beta program, instances used direct addressing. This addressing scheme used the same address for internal and external access. This approach is being deprecated; the documentation does not discuss this addressing scheme.

#### **Private (RFC 1918) Addresses**

All Amazon EC2 instances are allocated a private address by DHCP. These addresses are from a range defined in <u>"RFC 1918 - Address Allocation for Private Internets"</u>, are only routable within Amazon EC2, and are used for communication between instances.

This private address is associated exclusively with the instance for its lifetime and is only returned to Amazon EC2 when the instance terminates.

Always use the internal address when you are communicating between Amazon EC2 instances. This ensures that your network traffic follows the highest bandwidth, lowest cost, and lowest latency path through our network.

#### To determine your IP address

- 1. Connect to the instance.
- 2. Enter one of the following:

```
PROMPT> ifconfig eth0

PROMPT> curl http://169.254.169.254/latest/meta-data/local-ipv4
```

The second option refers to the instance data (see **Instance Metadata**).

#### **Internal DNS Name**

Each instance is provided an internal DNS name in the form domU-12-31-35-00-35-F3.compute-1.internal. It will resolve to the private IP address of the instance from within Amazon EC2; it will not resolve outside of Amazon EC2.

#### **Public Addresses**

At launch, a public address is also associated with each Amazon EC2 instance using Network Address Translation (NAT). For more information about NAT, see "RFC 1631: The IP Network Address Translator (NAT)".

This public address is associated exclusively with the instance for its lifetime and is only returned to Amazon EC2 when the instance terminates.

You must use this public address to access your instance outside of Amazon EC2 and you should distribute this address to clients (or through a DNS name that maps to it).



#### **Important**

Amazon EC2 instances cannot access other instances through their public NAT IP address; the private address of the instance in the new NAT environment must be used.

From your instance, you can determine your public IP address by referring to the instance data. For example:

PROMPT> curl http://169.254.169.254/latest/meta-data/public-ipv4

#### **Public DNS**

Each instance is provided an external DNS name in the form ec2-72-44-45-204.compute-1.amazonaws.com. This DNS name resolves to the public IP address of the instance outside the Amazon EC2 network and the private IP address from within Amazon EC2 network.



#### (1) Important

Amazon EC2 instances cannot access instances in the new NAT environment using their public NAT IP address; they must use the private addresses of the instances in the new NAT environment.

## **Network Security**

The Amazon EC2 service allows you to dynamically add and remove instances. However, this flexibility can complicate firewall configuration and maintenance which traditionally relies on IP addresses, subnet ranges or DNS host names as the basis for the firewall rules.

The Amazon EC2 firewall allows you to assign your instances to user-defined groups and define firewall rules for these groups. As instances are added or removed, the appropriate rules are enforced. Similarly, if you change a rule for a group, the changes are automatically applied to all members of the group.

### **Notes**

- Defining firewall rules in terms of groups is flexible enough to allow you to implement functionality equivalent to a VLAN.
- In addition to the distributed firewall, you can maintain your own firewall on any of your instances. This can be useful if you have specific requirements not met by the Amazon EC2 distributed firewall.

# **Concepts**

## **Security Groups**

A security group is a named collection of access rules. These access rules specify which ingress (i.e., incoming) network traffic should be delivered to your instance. All other ingress traffic will be discarded.

You can modify rules for a group at any time. The new rules are automatically enforced for all running instances and instances launched in the future.



You can create up to 100 rules per group.

## **Group Membership**

When you launch an AMI instance, you can assign it to as many groups as you like.

If no groups are specified, the instance is assigned to the default group. By default, this group allows all network traffic from other members of this group and discards traffic from other IP addresses and groups. If this does not meet your needs, you can modify the rule settings of the default group.

## **Group Access Rights**

The access rules define source based access either for named security groups or for IP addresses (i.e., CIDR-based rules). For CIDR-based rules, you can also specify the protocol and port range (or ICMP type/code).

## **Examples**

This section provides two examples of how to use the Amazon EC2 firewall.



These examples use the **command line tools**. You can achieve the same results using the **SOAP API**.

## **Default Group**

This example shows Albert modifying the default group to meet his security needs.

1. Albert launches a copy of his favorite public AMI:

```
$ ec2-run-instances ami-eca54085
INSTANCE i-cfd732a6 ami-eca54085 pending 0 m1.small 2007-07-11T16:4
```

2. After a little wait for image launch to complete. Albert, who is a cautious type, checks the access rules of the default group:

```
$ ec2-describe-group default
GROUP 598916040194 default default group
PERMISSION default ALLOWS all FROM US
```

Albert notices that it only accepts ingress network connections from other members of the default group for all protocols and ports.

3. Albert, being paranoid as well as cautious, port scans his instance:

```
$ nmap -P0 -p1-100 domU-12-31-33-00-01-56.usma1.compute.amazonaws.compute.governmap 3.81 ( http://www.insecure.org/nmap/ ) at 2006-08-07 All 100 scanned ports on domU-12-31-33-00-01-56.usma1.compute.amazo Nmap finished: 1 IP address (1 host up) scanned in 31.008 seconds
```

4. Albert decides he should be able to SSH into his instance, but only from his own machine:

```
$ ec2-authorize default -P tcp -p 22 -s 192.168.1.130/32
GROUP default
PERMISSION default ALLOWS tcp 22 22 FRO
```

5. Albert repeats the port scan:

```
$ nmap -P0 -p1-100 domU-12-31-33-00-01-56.usma1.compute.amazonaws.co
Starting nmap 3.81 ( http://www.insecure.org/nmap/ ) at 2006-08-07
Interesting ports on domU-12-31-33-00-01-56.usma1.compute.amazonaws
(The 99 ports scanned but not shown below are in state: filtered)
PORT STATE SERVICE
22/tcp open ssh
Nmap finished: 1 IP address (1 host up) scanned in 32.705 seconds
```

Albert is happy (or at least less paranoid).

#### Three-Tier Web Service

Mary wishes to deploy her public, fault tolerant, three-tier web service in Amazon EC2. Her grand plan is to have her web tier start off executing in seven instances of ami-fba54092, her application tier executing in twenty instances of ami-e3a5408a, and her multi-master database in two instances of ami-f1a54098. She's concerned about the security of her subscriber database, so she wants to restrict network access to her middle and back tier machines. When the traffic to her site increases over the holiday shopping period, she adds additional instances to her web and application tiers to handle the extra load.

1. First, Mary creates a group for her Apache web server instances and allows HTTP access to the world.

```
$ ec2-add-group apache -d "Mary's Apache group"
        apache Mary's Apache group
$ ec2-describe-group apache
        598916040194
GROUP
                        apache Mary's Apache group
$ ec2-authorize apache -P tcp -p 80 -s 0.0.0.0/0
GROUP
                apache
PERMISSION
                         apache ALLOWS tcp
                                                 80
                                                          80
                                                                  FR<sub>0</sub>
$ ec2-describe-group apache
                         apache Mary's Apache group
        598916040194
PERMISSION
                598916040194
                                 apache ALLOWS tcp
                                                          80
                                                                  80
```

2. Mary launches seven instances of her web server AMI as members of the apache group:

```
$ ec2run ami-fba54092 -n 7 -g apache
INSTANCE i-cfd732a6 ami-fba54092 pending 0 ml.small 2007-07-11T16:4
INSTANCE i-cfd732a7 ami-fba54092 pending 0 ml.small 2007-07-11T16:4
INSTANCE i-cfd732a8 ami-fba54092 pending 0 ml.small 2007-07-11T16:4
INSTANCE i-cfd732a9 ami-fba54092 pending 0 ml.small 2007-07-11T16:4
INSTANCE i-cfd732aa ami-fba54092 pending 0 ml.small 2007-07-11T16:4
INSTANCE i-cfd732ab ami-fba54092 pending 0 ml.small 2007-07-11T16:4
INSTANCE i-cfd732ac ami-fba54092 pending 0 ml.small 2007-07-11T16:4
```

```
$ ec2din i-cfd732a6
RESERVATION r-0592776c 598916040194
INSTANCE i-cfd732a6 ami-fba54092 domU-12-31-33-00
m1.small 2007-07-11T16:40:44+0000
```

3. Being as paranoid as Albert, Mary does a port scan to confirm the permissions she just configured:

4. Mary verifies her web server can be reached:

```
$ telnet domU-12-31-33-00-04-16.usmal.compute.amazonaws.com 80
Trying 216.182.231.20...
Connected to domU-12-31-33-00-04-16.usmal.compute.amazonaws.com (21
Escape character is '^]'.
```

Mary can reach her web server.

5. Mary creates a separate group for her application server:

```
$ ec2-add-group appserver -d "Mary's app server"
GROUP appserver Mary's app server
```

6. Mary starts twenty instances as members of appserver group:

```
s ec2run ami-e3a5408a -n 20 -g appserver
```

7. Mary grants network access between her web server group and the application server group:

```
$ ec2-authorize appserver -o apache -u 495219933132
GROUP appserver
PERMISSION appserver ALLOWS all FROM
```

8. Mary verifies access to her app server is restricted by port scanning one of the application servers:

```
$ nmap -P0 -p1-100 domU-12-31-33-00-03-D1.usmal.compute.amazonaws.compute.governmap 3.81 ( http://www.insecure.org/nmap/ ) at 2006-08-07
All 100 scanned ports on domU-12-31-33-00-03-D1.usmal.compute.amazo
Nmap finished: 1 IP address (1 host up) scanned in 31.008 seconds
```

- 9. Mary confirms that her web servers have access to her application servers:
  - a. She (temporarily) grants SSH access from her workstation to the web server group:

```
s ec2-authorize apache -P tcp -p 22 -s 192.168.1.130/32
```

b. She logs in to one of her web servers and connects to an application server on TCP port 8080:

```
$ telnet domU-12-31-33-00-03-D1.usmal.compute.amazonaws.com 808 Trying 216.182.228.12... Connected to domU-12-31-33-00-03-D1 .usmal.compute.amazonaws.cc Escape character is '^]'
```

c. Satisfied with the setup, she revokes SSH access to the web server group:

```
$ ec2-revoke apache -P tcp -p 22 -s 192.168.1.130/32
```

10. Mary repeats these steps to create the database server group and to grant access between the application server and database server groups.

# **Using the APIs**

### **Topics**

- <u>Using the SOAP API</u>
- <u>Using the Query API</u>

This section provides an overview of the SOAP and Query APIs.

## **Using the SOAP API**

### **Topics**

- WSDL and Schema Definitions
- Programming Language Support in Amazon EC2
- Request Authentication
- The Response Structure
- Web Services References

#### **WSDL** and Schema Definitions

The Amazon EC2 web service can be accessed using the SOAP web services messaging protocol. This interface is described by a Web Services Description Language (WSDL) document which defines the operations and security model for the service. The WSDL references an XML Schema document which strictly defines the data types that might appear in SOAP requests and responses. For more information on WSDL and SOAP, see <a href="Web Services References">Web Services References</a>.

All schemas have a version number (the latest is 2007-08-29). The version number appears in the URL of a schema file, and in a schema's target namespace. This makes upgrading easy by differentiating requests based on the version number.



In addition to the latest version, the service will support the older versions for some time, allowing customers plenty of time to upgrade.

The Amazon EC2 services API WSDL is available from the web at 'http://ec2.amazonaws.com/doc/<version>/ec2.wsdl' where *version* is the version of the API. At the time this document was released, the current API version was 2007-08-29, which is available at <a href="http://ec2.amazonaws.com/doc/2007-08-29/AmazonEC2.wsdl">http://ec2.amazonaws.com/doc/2007-08-29/AmazonEC2.wsdl</a>

## **Programming Language Support in Amazon EC2**

Since the SOAP requests and responses in the Amazon EC2 Web Service follow current standards, any programming language with the appropriate library support can be used. Languages known to have this support include C++, C#, Java, Perl, Python and Ruby.

## **Request Authentication**

The Amazon EC2 web service complies with the current WS-Security standard, requiring SOAP request messages to be hashed and signed for integrity and non-repudiation. WS-Security defines profiles which are used to implement various levels of security. Amazon EC2 secure SOAP messages use the BinarySecurityToken profile, consisting of an X.509 certificate with an RSA public key.

The following is the content of an insecure RunInstances operation:

To secure the request, we add the BinarySecurityToken element. The Java libraries we supply rely on the Apache Axis project for XML security, canonicalization, and SOAP support. The Sun Java Web Service Developer's Pack supplies libraries of equivalent functionality.

The secure version of the request begins with the following:

```
<ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmlds</pre>
        <ds:Reference URI="#id-17984263">
          <ds:Transforms>
            <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc</pre>
          </ds:Transforms>
          <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsi</pre>
          <ds:DigestValue>0pjZ1+TvgPf6uG7o+Yp3l2YdGZ4=</ds:DigestValue</pre>
        </ds:Reference>
        <ds:Reference URI="#id-15778003">
          <ds:Transforms>
            <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc</pre>
          </ds:Transforms>
          <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsi</pre>
          <ds:DigestValue>HhRbxBBmc200348f8nLNZyo4A0M=</ds:DigestValue</pre>
        </ds:Reference>
      </ds:SignedInfo>
      <ds:SignatureValue>bmVx24Qom4kd9QQtclxWIlgLk4QsQBPaKESi79x479xgl
      jjHKZKEQRCOlLVy0Dn5ZL1RlMHsv+0zJzzvIJFTq3LQKNrzJzsNe</ds:Signati
      <ds:KeyInfo Id="KeyId-17007273">
        <wsse:SecurityTokenReference</pre>
            xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-20
          <wsse:Reference URI="#CertId-1064304"</pre>
                           ValueType="http://docs.oasis-open.org/wss/20
          </wsse:Reference>
        </wsse:SecurityTokenReference>
      </ds:KeyInfo>
    </ds:Signature>
    <wsu:Timestamp</pre>
        xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401
      <wsu:Created>2006-06-09T10:57:35Z</wsu:Created>
      <wsu:Expires>2006-06-09T11:02:35Z</wsu:Expires>
    </wsu:Timestamp>
  </wsse:Security>
</SOAP-ENV:Header>
```

If you are matching this against requests generated by Amazon EC2 supplied libraries, or those of another vendor, the following are the most important elements:

- **BinarySecurityToken**—Contains the X.509 certficiate in base64 encoded PEM format.
- **Signature**—Contains an XML digital signature created using the canonicalization, signature algorithm, and digest method.
- **Timestamp**—Requests to Amazon EC2 are valid within 5 minutes of this value. This helps prevent replay attacks.

## The Response Structure

In response to a request, the Amazon EC2 web service returns an XML data structure that conforms to an XML schema defined as part of the Amazon EC2 WSDL. The structure of a XML response is specific to the associated request. In general, the response data types are named according to the operation performed and whether the data type is a container (can have children). Examples of containers include groupSet for security groups and instancesSet for instances. Item elements are children of containers and their contents vary according to the container's role.

#### An example response is:

```
<RunInstancesResponse xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
  <reservationId>r-47a5402e</reservationId>
  <ownerId>UYY3TLBUXIEON5NQVUUX60MPWBZIQNFM</ownerId>
  <groupSet>
    <item>
      <groupId>default
    </item>
 </groupSet>
  <instancesSet>
    <item>
      <instanceId>i-2ba64342</instanceId>
      <imageId>ami-60a54009</imageId>
      <instanceState>
        <code>0</code>
    <name>pending</name>
      </instanceState>
      <dnsName></dnsName>
    </item>
    <item>
      <instanceId>i-2bc64242</instanceId>
      <imageId>ami-60a54009</imageId>
      <instanceState>
        <code>0</code>
    <name>pending</name>
      </instanceState>
      <dnsName>domU-13-35-33-00-00-5C.dc2.compute.amazonaws.com</dnsName</pre>
    </item>
    <item>
      <instanceId>i-2be64332</instanceId>
      <imageId>ami-60a54009/imageId>
      <instanceState>
```

```
<code>0</code>
  <name>pending</name>
    </instanceState>
    <dnsName>domU-12-34-28-00-00-5C.dc2.compute.amazonaws.com</dnsName
    <keyName>example-key-name</keyName>
        <instanceType>m1.small</instanceType>
        <launchTime>2007-08-07T11:54:42.000Z</launchTime>
        </item>
        </instancesSet>
</RunInstancesResponse>
```

## **Web Services References**

- Web Service Description Language (WSDL)
- WS-Security BinarySecurityToken Profile

## **Using the Query API**

### **Topics**

- Query Parameters
- Query API Authentication
- Example Request

HTTP Query-based requests are HTTP requests that use the HTTP verb GET or POST and a Query parameter named Action or Operation. Action is used throughout this documentation, although Operation is supported for backward compatibility with other AWS Query APIs.

## **Query Parameters**

Each Query request must include some common parameters to handle authentication and selection of an action. For more information, see <a href="Common Query Parameters">Common Query Parameters</a>.

Some operations take lists of parameters. These lists are specified using the param.n notation. Values of n are integers starting from 1.

## **Query API Authentication**

Every request to Amazon EC2 must contain a request signature. A request signature is calculated by constructing a string and then calculating an RFC 2104-compliant HMAC-SHA1 hash, using the Secret AWS Access Key as the key. For more information, see <a href="http://www.fags.org/rfcs/rfc2104.html">http://www.fags.org/rfcs/rfc2104.html</a>.

The following are the basic steps used to authenticate requests to AWS. This assumes the developer is registered with AWS and has an Access Key ID and Secret Access Key.

#### **Query Authentication**

- 1 The sender constructs a request to AWS.
- 2 The sender calculates the request signature, a Keyed-Hashing for Message Authentication Code (HMAC) with a SHA-1 hash function, as defined in the next section of this topic.
- 3 The sender of the request sends the request data, the signature, and Access Key ID (the keyidentifier of the Secret Access Key used) to AWS.
- 4 AWS uses the Access Key ID to look up the Secret Access Key.
- 5 AWS generates a signature from the request data and the Secret Access Key using the same algorithm used to calculate the signature in the request.
- 6 If the signatures match, the request is considered to be authentic. If the comparison fails, the request is discarded, and AWS returns an error response.



If a request contains a *Timestamp* parameter, the signature calculated for the request expires 15 minutes its value. If a request contains an *Expires* parameter, the signature expires at the time specified by the *Expires* parameter.

### To calculate the request signature

- 1. Based on the API (Query/SOAP/REST) you are using, construct a string.
  - a. Sort the query parameters (not URL-encoded) without using casesensitively.
  - b. Concatenate the parameter names and values without the initial? or the separating & and = characters.

- 2. Compute an RFC 2104 compliant HMAC using the Secret AWS Access Key as the "key".
- 3. Convert the value to base64.
- 4. Include the value as the value of the *Signature* parameter in the request.

For example, below is a Query string (linebreaks added for clarity):

```
?Action=DescribeImages
&AWSAccessKeyId=10QMXFEV71ZS32XQFTR2
&SignatureVersion=1
&Timestamp=2006-12-08T07%3A48%3A03Z
&Version=2007-01-03
```

For the above Query string, you would calculate the HMAC signature over the following string:

```
ActionDescribeImagesAWSAccessKeyId10QMXFEV71ZS32XQFTR2SignatureVersion17
```

Using the string above and the secret key DMADSSfPfdaDjbK+RRUhS/aDrjsiZadgAUm8gRU2 the base64 encoded signature is as follows:

```
GjH3941IBe6qsgQu+k7FpCJjpnc=
```

The following is a Java code sample to compute the signature from the string and the private key.

```
@param data
           The data to be signed.
     * @param key
           The signing key.
      @return
           The base64-encoded RFC 2104-compliant HMAC signature.
     * @throws
           java.security.SignatureException when signature generation fa
     */
    public static String calculateRFC2104HMAC(String data, String key)
        throws java.security.SignatureException
    {
        String result;
        try {
            // get an hmac shal key from the raw key bytes
            SecretKeySpec signingKey = new SecretKeySpec(key.getBytes(),
                                                          HMAC SHA1 ALGOF
            // get an hmac_shal Mac instance and initialize with the sig
            Mac mac = Mac.getInstance(HMAC SHA1 ALGORITHM);
            mac.init(signingKey);
            // compute the hmac on input data bytes
            byte[] rawHmac = mac.doFinal(data.getBytes());
            // base64-encode the hmac
            result = Base64.encodeBytes(rawHmac);
        catch (Exception e) {
            throw new SignatureException("Failed to generate HMAC : " +
        }
        return result;
    }
}
```

#### Note

You must import a base 64 encoder to perform the last step above.

## **Example Request**

Here is a complete example request, including all required parameters:

? AWS Access Key Id = 10 QMX FEV 71 ZS 32 XQFTR 2 & Action = Describe Images & Signature Version = Signa

## **API Reference**

### **Topics**

- API Conventions
- API Versioning
- API Error Codes
- Common Data Types
- Amazon EC2 SOAP API
- Amazon EC2 Query API

Amazon EC2 provides two APIs: <u>SOAP</u> and <u>Query</u>, which allow you to launch and control instances from your own applications.

This section discusses the operations available in the Amazon EC2 APIs, their semantics, and their required parameters. Examples of requests and responses are also provided.



The same XML body is returned in both the Query API and SOAP API.

Before using the API, we recommend that you familiarize yourself with their <u>conventions</u>.

## **API Conventions**

## Overview

This section describes Amazon EC2 API conventions.

### **Actions**

Actions encapsulate the possible interactions with Amazon EC2. These can be viewed as remote procedure calls and consist of a request and response message pair. Requests must be signed, allowing Amazon EC2 to <u>authenticate the caller</u>. For clarity, the sample requests and responses illustrating each of the operations described in this reference are not signed.

### **Data Types and the Amazon EC2 WSDL**

The current version of the Amazon EC2 WSDL is available at: <a href="http://ec2.amazonaws.com/doc/2007-08-29/AmazonEC2.wsdl">http://ec2.amazonaws.com/doc/2007-08-29/AmazonEC2.wsdl</a>. Some libraries can generate code directly from the WSDL. Other libraries require a little more work on your part.

Values provided as parameters to the various operations must be of the indicated type. Standard XSD types (like string, boolean, int) are prefixed with xsd:. Complex types defined by the Amazon EC2 WSDL are prefixed with ec2:.

Parameters that consist of lists of information are defined within our WSDL to require <info> tags around each member. Throughout the API, type references for parameters that accept such a list of values are specified using the notation type[] The type referred to in these instances is the type *nested within the* <*info> tag* (for Amazon EC2 types this is defined in the WSDL).

For example, the <imagesSet> element in the following XML snippet is of type xsd:string[]:

And the <instancesSet> element in the following XML snippet is of type ec2:RunInstanceItemType[]:

```
<minCount>5</minCount>
    <maxCount>20</maxCount>
    </item>
</instancesSet>
```

## **API Versioning**

Because features and changes can introduce incompatible API changes, all Amazon EC2 API updates are versioned. By including a version in the request, clients receive responses they can process.

Each API revision is assigned a version in date form (the current API version is 2007-08-29). This version is included in the request as part of the document namespace when using our SOAP API and as a Version parameter when using our Query API. The response that Amazon EC2 returns honors the version included in the request.

SOAP clients that retrieve the Amazon EC2 WSDL at runtime and generate their requests dynamically using that WSDL should reference the WSDL for the version of the API that the client was developed against. This ensures that the client software continues to work even if backwards incompatible API changes are introduced. The WSDL for each supported API version is available from the following URI:

http://ec2.amazonaws.com/doc/<api-version>/AmazonEC2.wsdl

The WSDL for latest version of our API is available from the following URI:

http://ec2.amazonaws.com/doc/AmazonEC2.wsdl



The WSDL should be treated as a moving target as it will always map to the latest release of the Amazon EC2 SOAP API. If your software depends on retrieving the WSDL at runtime, we strongly recommend you reference the specific version of the WSDL you are developing against.

## **API Error Codes**

### **Overview**

There are two types of error codes: client and server.

Client error codes suggest that the error was caused by something the client did, such as an authentication failure or an invalid AMI identifier. In the SOAP API, These error codes are prefixed with Client. For example: Client.AuthFailure. In the Query API, these errors are accompanied by a 400-series HTTP response code.

Server error codes suggest a server-side issue caused the error and should be reported. In the SOAP API, these error codes are prefixed with Server. For example: Server.Unavailable. In the Query API, these errors are accompanied by a 500-series HTTP response code.

## **Summary of Client Error Codes**

Error Code	Description	Notes
AuthFailure	User not authorized.	You might be trying to run an AMI for which you do not have permission.
InvalidManifest	Specified AMI has an unparsable Manifest.	
InvalidAMIID.Malformed	Specified AMI ID is not valid.	
InvalidAMIID.NotFound	Specified AMI ID does not exist.	
InvalidAMIID.Unavailable	Specified AMI ID has been deregistered and is no longer available.	
InvalidInstanceID.Malformed	Specified instance ID is not valid.	
InvalidInstanceID.NotFound	Specified instance ID does not exist.	
InvalidKeyPair.NotFound	Specified key pair name does not exist.	
InvalidKeyPair.Duplicate	Attempt to create a duplicate key pair.	
InvalidGroup.NotFound	Specified group name does not exist.	
InvalidGroup.Duplicate	Attempt to create a duplicate group.	
InvalidGroup.InUse	Specified group can not be deleted because it is in use.	
InvalidGroup.Reserved	Specified group name is a reserved name.	
InvalidParameterValue	The value supplied for a parameter was invalid.	Requests that could cause this error include (for example) supplying an invalid image attribute to the DescribeImageAttribute request or an invalid version or encoding value for the userData in a

		RunInstances request.
InvalidPermission.Duplicate	Attempt to authorize a permission that has already been authorized.	
InvalidPermission.Malformed	Specified permission is invalid.	
InvalidReservationID.Malformed	Specified reservation ID is invalid.	
InvalidReservationID.NotFound	Specified reservation ID does not exist.	
InstanceLimitExceeded	User has max allowed concurrent running instances.	Each user has a concurrent running instance limit. For new users during public beta, this limit is 20.
InvalidParameterCombination	RunInstances was called with minCount and maxCount set to 0 or minCount > maxCount.	
InvalidUserID.Malformed	The user ID is neither in the form of an AWS account ID or one of the special values accepted by the owner or executableBy flags in the DescribeImages call.	
InvalidAMIAttributeItemValue	The value of an item added to, or removed from, an image attribute is invalid.	If you are specifying a userId, check that it is in the form of an AWS account ID.
UnknownParameter	An unknown or unrecognized parameter was supplied.	Requests that could cause this error include supplying a misspelled parameter or a parameter that is not supported for the specified API version.

## **Summary of Server Error Codes**

Error Code	Description	Notes
InternalError	Internal Error.	This error should not occur. If it does, please try to reproduce it and let us know by posting a message on the <u>AWS forums</u> .
InsufficientInstanceCapacity	Not enough available instances to satisfy your minimum request.	Reduce the number of instances in your request or wait for additional capacity to become available.
Unavailable	The server is overloaded and cannot handle the request.	

## **Common Data Types**

The Amazon EC2 API contains several data types that various operations use. This section describes each data type in detail.

Since both the Query and SOAP APIs return the same XML body, the data types described in the WSDL are used in both.

# ${\bf Describe Images Response Item Type}$

The DescribeImagesResponseItemType data type.

Operations that use this data type include:

• <u>DescribeImages</u>

The following table describes the elements contained in DescribeImagesResponseItemType.

Name	Description
imageId	Unique ID of the AMI described. Type: xsd:string
imageState	Current state of the AMI.  If the operation returns available, the image is successfully registered and available for launching  If the operation returns deregistered, the image is deregistered and no longer available for launching  Type: xsd:string
imageOwnerId	AWS Access Key ID of the image owner. Type: xsd:string
isPublic	Returns true if this image has public launch permissions. Returns false if it only has implicit and explicit launch permissions.  Type: xsd:boolean
productCodes	Product codes associated with this image.  Type: ec2:ProductCodeItemType[]

# ${\bf Describe Key Pairs Response Item Type}$

The DescribeKeyPairsResponseItemType data type.

- <u>DeleteKeypair</u>
- <u>DescribeKeypairs</u>

The following table describes the elements contained in DescribeKeyPairsResponseItemType.

Name	Description
keyName	The user supplied name for this key pair.  Type: xsd:string
keyFingerprint	A fingerprint for the private key of this key pair. This is computed as the SHA-1 digest of the DER encoded form of the private key.  Type: xsd:string

# ${\bf EmptyElementType}$

The EmptyElementType data type.

- <u>ResetImageAttribute</u>
- <u>DescribeImageAttribute</u>

The empty element has no contents.

# GroupSetType

The GroupSetType data type.

Operations that use this data type include:

• RunInstances

The following table describes the elements contained in GroupSetType.

Name	Description
groupId	Name of a security group. Type: xsd:string

# In stance State Type

The InstanceStateTypedata type.

- RunInstances
- <u>DescribeInstances</u>
- <u>TerminateInstances</u>

The following table describes the elements contained in InstanceStateType.

Name	Description
code	A 16-bit unsigned integer. The high byte is an opaque internal value and should be ignored. The low byte is set based on the state represented:
	• 0: pending
	• 16: running
	• 32: shutting-down
	• 48: terminated
	Type: xsd:int
state	The current state of the instance.
	<ul> <li>pending: the instance is in the process of being launched</li> </ul>
	• running: the instance launched (although the boot process might not be completed)
	• shutting-down: the instance started shutting down
	• terminated: the instance terminated
	Type: xsd:string

# **IpPermissionType**

The IpPermissionType data type.

- <u>AuthorizeSecurityGroupIngress</u>
- <u>DescribeSecurityGroups</u>
- RevokeSecurityGroupIngress

The following table describes the elements contained in IpPermissionType.

Name	Description
ipProtocol	IP protocol. Type: xsd:string
fromPort	Start of port range for the TCP and UDP protocols, or an ICMP type number. An ICMP type number of -1 indicates a wildcard (i.e., any ICMP type number). Type: xsd:int
toPort	End of port range for the TCP and UDP protocols, or an ICMP code. An ICMP code of -1 indicates a wildcard (i.e., any ICMP code).  Type: xsd:int
groups	List of security group and user ID pairs.  Type: ec2:UserIdGroupPairType[]
ipRanges	List of CIDR IP range specifications. Type: xsd:string

# Launch Permission Item Type

The LaunchPermissionItemType data type.

- ModifyImageAttribute
- <u>DescribeImageAttribute</u>

The following table describes the elements contained in LaunchPermissionItemType.

Name	Description	Required
group	A launch permission for a group. Currently only all is supported, which gives public launch permissions.  Type: xsd:string	Choice between group and userId
userId	A launch permission for a user. userId is an AWS account ID. Type: xsd:string	Choice between group and userId

# Launch Permission Operation Type

The LaunchPermissionOperationType data type.

Operations that use this data type include:

• ModifyImageAttribute

The following table describes the elements contained in LaunchPermissionOperationType.

Name	Description	Required
add	Adds launch permissions.  Type: ec2:LaunchPermissionItemType[]	Choice between add and remove
remove	Removes launch permissions.  Type: ec2:LaunchPermissionItemType[]	Choice between add and remove

# ${\bf Product Code Item Type}$

 $The\ Product Code Item Type\ data\ type.$ 

- ModifyImageAttribute
- <u>DescribeImageAttribute</u>

The following table describes the elements contained in ProductCodeItemType.

Name	Description	Required
productCode	A product code. Type: xsd:string	Yes

# ${\bf Product Instance Response Item Type}$

The ProductInstanceResponseItemType data type.

Operations that use this data type include:

• <u>ConfirmProductInstance</u>

The following table describes the elements contained in ProductInstanceResponseItemType.

Name	Description
productCode	The product code attached to the instance that matches one of the product codes in the ConfirmProductInstance request.  Type: xsd:string
instanceId	Unique ID of the instance. Type: xsd:string
ownerId	The account ID of the owner of the instance. Type: xsd:string

# ${\bf Reservation Info Type}$

The ReservationInfoType data type.

- RunInstances
- <u>DescribeInstances</u>

The following table describes the elements contained in ReservationInfoType.

Name	Description
reservationId	Unique ID of the reservation described. Type: xsd:string
ownerId	AWS Access Key ID of the user who owns the reservation. Type: xsd:string
groupSet	Set of security groups these instances were launched in. Type: <a href="mailto:ec2:GroupSetType">ec2:GroupSetType</a> []
instancesSet	Information about instances started.  Type: ec2:RunningInstancesItemType[]

# ${\bf RunInstance Item Type}$

The RunInstanceItemType data type.

# **Relevant Operations**

Operations that use this data type include:

• RunInstances

The following table describes the elements contained in RunInstanceItemType.

Name	Description
imageId	Unique ID of a machine image, returned by a call to <a href="RegisterImage">RegisterImage</a> . Type: xsd:string
minCount	Minimum number of instances to launch. If minCount is more than Amazon EC2 can launch, no instances are launched at all.  Type: xsd:int
maxCount	Maximum number of instances to launch. If maxCount is more than Amazon EC2 can launch, the largest possible number above minCount will be launched instead. Type: xsd:int
keyName	The name of the key pair. Type: xsd:string

# ${\bf Running Instances Item Type}$

The RunningInstancesItemType data type.

# **Relevant Operations**

Operations that use this data type include:

• RunInstances

The following table describes the elements contained in RunningInstancesItemType.

Name	Description
instanceId	Unique ID of the instance launched. Type: xsd:string
imageId	Image ID of the AMI used to launch the instance. Type: xsd:string
instanceState	The current state of the instance.  • pending: the instance is in the process of launching  • running: the instance launched (although it the boot process might not be complete)  • shutting-down: the instance is shutting down  • terminated: the instance terminated  Type: ec2:InstanceStateType
privateDnsName	The private DNS name assigned to the instance. This DNS name can only be used inside the Amazon EC2 network. This element remains empty until the instance enters a running state. For more information, see <a href="Instance Addressing and Network Security">Instance Addressing and Network Security</a> . Type: xsd:string
dnsName	The public DNS name assigned to the instance. This DNS name is contactable from outside the Amazon EC2 network. This element remains empty until the instance enters a running state. For more information, see <a href="Instance Addressing and Network Security">Instance Addressing and Network Security</a> . Type: xsd:string
reason	Optional. Reason for the most recent state transition. This might be an empty string. Type: xsd:string
keyName	Optional. If this instance was launched with an associated key pair, this displays the key pair name.  Type: xsd:string
amiLaunchIndex	Optional. The AMI launch index, which can be used to find this instance within the launch group. For more information, see <a href="Instance Metadata">Instance Metadata</a> . Type: xsd:string
productCodes	Optional. Product codes attached to this instance.  Type: ec2:ProductCodeItemType[]
instanceType	The instance type. For more information on instance types, see <u>Selecting Instance Types</u>

	Type: xsd:string
launchTime	The time the instance launched.  Type: xs:dateTime

# **SecurityGroupItemType**

The SecurityGroupItemType data type.

## **Relevant Operations**

Operations that use this data type include:

• <u>DescribeSecurityGroups</u>

The following table describes the elements contained in SecurityGroupItemType.

Name	Description	
ownerId	AWS Access Key ID of the owner of the security group. Type: xsd:string	
groupName	Name of the security group. Type: xsd:string	
groupDescription	Description of the security group. Type: xsd:string	
ipPermissions	Set of IP permissions associated with the security group Type: <a href="mailto:ec2:IpPermissionType">ec2:IpPermissionType</a> []	

# Terminate Instances Response Info Type

 $The\ TerminateInstancesResponseInfoType\ data\ type.$ 

# **Relevant Operations**

Operations that use this data type include:

• <u>TerminateInstances</u>

The following table describes the elements contained in TerminateInstancesResponseInfoType.

Name	Description
	Instance ID returned from previous call to <u>RunInstances</u> . Type: xsd:string

# **UserDataType**

The UserDataType data type.

# **Relevant Operations**

Operations that use this data type include:

• RunInstances

The following table describes the elements contained in UserDataType.

Name	Description
data	The user data. Type: xsd:string

#### **Notes**

- The version and encoding attributes are required.
- The user data is base64-encoded as described in <a href="RFC3548">RFC3548</a> with the following additional restrictions:
  - Implementations MUST NOT add line feeds to encoded data.
  - Implementations MUST pad the end of the encoded data with '=' if required.
  - Implementations MUST ignore characters in the encoded stream that are not in the encoding alphabet. This differs from RFC3548, but provides more leeway for clients.
  - Implementation MUST use the encoding alphabet in table 1 of RFC3548 (i.e. A-Za-z0-9+/).
  - Implementation MUST follow the user data size limit before base64 encoding.

# UserIdGroupPairType

The UserIdGroupPairType data type.

## **Relevant Operations**

Operations that use this data type include:

- <u>AuthorizeSecurityGroupIngress</u>
- <u>DescribeSecurityGroups</u>
- RevokeSecurityGroupIngress

The following table describes the elements contained in UserIdGroupPairType.

Name	Description
userId	AWS Access Key ID of a user. Type: xsd:string
groupName	Name of a security group. Type: xsd:string

## **Amazon EC2 SOAP API**

The Amazon EC2 API consists of web service operations for every task the service can perform. This section describes each operation in detail.

# **List of Operations by Function**

#### **Images**

- RegisterImage
- DescribeImages
- <u>DeregisterImage</u>

#### **Instances**

- RunInstances
- <u>DescribeInstances</u>
- <u>TerminateInstances</u>
- ConfirmProductInstance

#### **Key Pairs**

- <u>CreateKeyPair</u>
- <u>DescribeKeyPairs</u>
- <u>DeleteKeyPair</u>

#### **Image Attributes**

- ModifyImageAttribute
- <u>DescribeImageAttribute</u>
- ResetImageAttribute

#### **Security Groups**

• CreateSecurityGroup

- <u>DescribeSecurityGroups</u>
- <u>DeleteSecurityGroup</u>
- <u>AuthorizeSecurityGroupIngress</u>
- RevokeSecurityGroupIngress

# **AuthorizeSecurityGroupIngress**

The AuthorizeSecurityGroupIngress operation adds permissions to a security group.

Permissions are specified by the IP protocol (TCP, UDP or ICMP), the source of the request (by IP range or an Amazon EC2 user-group pair), the source and destination port ranges (for TCP and UDP), and the ICMP codes and types (for ICMP).

Permission changes are propagated to instances within the security group as quickly as possible. However, depending on the number of instances, a small delay might occur.

## **Request Parameters**

The following table describes the request parameters for AuthorizeSecurityGroupIngress. Parameter names are case sensitive.

Name	Description	Required
userId	AWS Access Key ID. Type: xsd:string	Yes
groupName	Name of the group to modify. Type: xsd:string	Yes
ipPermissions	Set of permissions to add to the group.  Type: ec2:IpPermissionType[]	Yes

## **Response Elements**

The following table describes the default response tags included in AuthorizeSecurityGroupIngress responses.

Name	Description
return	true if permissions successfully added. Type: xsd:boolean

### **Sample Request**

```
<AuthorizeSecurityGroupIngress xmlns="http://ec2.amazonaws.com/doc/2007-</pre>
    <userId/>
    <groupName>WebServers
    <ipPermissions>
        <item>
            <ipProtocol>tcp</ipProtocol>
            <fromPort>80</fromPort>
            <toPort>80</toPort>
            <groups/>
            <ipRanges>
                <item>
                    <cidrIp>0.0.0.0/0</cidrIp>
                </item>
            </ipRanges>
        </item>
    </ipPermissions>
</AuthorizeSecurityGroupIngress>
```

## **Sample Response**

<AuthorizeSecurityGroupIngressResponse xmlns="http://ec2.amazonaws.com/c
<return>true</return>
</AuthorizeSecurityGroupIngressResponse>

## **Related Operations**

- <u>CreateSecurityGroup</u>
- <u>DescribeSecurityGroups</u>
- RevokeSecurityGroupIngress
- DeleteSecurityGroup

## **ConfirmProductInstance**

The ConfirmProductInstance operation returns true if the specified product code is attached to the specified instance. The operation returns false if the product code is not attached to the instance.

The ConfirmProductInstance operation can only be executed by the owner of the AMI. This feature is useful when an AMI owner is providing support and wants to verify whether a user's instance is eligible.

## **Request Parameters**

The following table describes the request parameters for ConfirmProductInstance. Parameter names are case sensitive.

Name	Description	Required
productCode	The product code to confirm. Type: xsdstring	Yes
instanceId	The instance for which to confirm the product code. Type: xsdstring	Yes

## **Response Elements**

The following table describes the default response tags included in ConfirmProductInstance responses.

Name	Description
return	True if the product code is attached to the instance, false if it is not.  Type: xsd:boolean
ownerId	The instance owner's account ID. Only present if the product code is attached to the instance.  Type: xsdstring

## **Sample Request**

## **Sample Response**

```
<ConfirmProductInstanceResponse xmlns="http://ec2.amazonaws.com/doc/2007
    <return>true</return>
        <ownerId>254933287430</ownerId>
        </ConfirmProductInstanceResponse>
```

# **Related Operations**

- <u>DescribeInstances</u>
- RunInstances

# CreateKeyPair

The CreateKeyPair operation creates a new 2048 bit RSA key pair and returns a unique ID that can be used to reference this key pair when <u>launching</u> new instances.

#### **Request Parameters**

The following table describes the request parameters for CreateKeyPair. Parameter names are case sensitive.

Name	Description	Required
keyName	A unique name for the key pair. Type: xsd:string	Yes

## **Response Elements**

The following table describes the default response tags included in CreateKeyPair responses.

Name	Description
keyName	The key pair name provided in the original request. Type: xsd:string
keyFingerprint	A SHA-1 digest of the DER encoded private key. Type: xsd:string
keyMaterial	An unencrypted PEM encoded RSA private key. Type: xsd:string

#### **Sample Request**

#### **Sample Response**

```
<CreateKeyPairResponse xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
  <keyName>example-key-name</keyName>
  <keyFingerprint>1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:
  <keyMaterial>----BEGIN RSA PRIVATE KEY-----
MIIEoQIBAAKCAQBuLFq5ujHrtmljnutSuo08Xe56LlT+HM8v/xkaa39EstM3/aFxTHqElQi.
HungXQ29VTc8rc1bW0lkdi230H5eqkMHGhvEwqa0HWASUMll4o3o/IX+0f2UcPoKC0VUR+j>
5AU52EOfanIn3Z08lFW7Edp5a3q4DhjGlUKToHVbicL5E+q45zfB95wIyywWZfeW/UUF3Lp(
ebIUlq1qTbHkLbCC2r7RTn8vpQWp47BGVYGtGSBMpTRP5hnbzzuqj3itkiLHjU39S2sJCJ07
i8BygR4s3mHKBj8l+ePQxG1kGbF6R4yg6sECmXn17MRQVX0DNHZbAgMBAAECggEAY1tsiUs]
91CXirkYGuVfLvLflXenxfI50mDFms/mumTqloH07tr0oriHDR5K7wMcY/YY5YkcXNo7mvU\
ZNUJs7rw9gZRTrf7LylaJ58k0cyajw8TsC4e4LPbFaHwS1d6K8rXh64o6WgW4SrsB6ICmr1
3wcfqt5ecIu4TZf00E9IHjn+2eRlsrjBde0Ri7KiUNC/pAG23I6MdD0FEQRcCSigCj+4/mci
SWS4dMbrpb9FNSIcf9dcLxVM7/6KxqJNfZc9XWzUw77Jq8x92Zd0fVhH0ux5IZC+UvSKWB4c
tE8C3p9bbU9VGyY5vLCAiIb4qQKBqQDLi024GXrIkswF32YtBBMuVqLGCwU9h9Hl09mKAc2r
jUE5IpzRjTedc9I2qiIMUTwtgnw42auSCzbUeYMURPtDqyQ7p6AjMujp9EPemcSV0K9vXYL(
xW9MC0dtV6iPkCN7q0qiZXPRKaFbWADp16p8UAIvS/a5XXk5jwKBqQCKkpHi2EISh1uRkhxl
iDCiK6JBRsMvpLbc0v5dKwP5alo1fmdR5PJaV2qvZSj5CYNpMAy1/EDNTY50SIJU+0KFmQby
rdLNLDL4+TcnT7c62/aH01ohYaf/VCbRhtLlBfgGo0c7+sAc8vmKkesnF7CqCEKDyF/dhrx\
qC0iZzzNAapayz1+JcVTwwEid6j9JqNXbBc+Z2YwMi+T0Fv/P/hwkX/ype0XnIUcw0Ih/Yt(
DQbsz7LcY1HqXiHKYNWNvXgww0+oiChjxvEkSdsTTIfnK4VSCvU9BxDbQHjdiNDJbL6oar91
rBYvChJZF7LvUH4YmVpHAoGAbZ2X7XvoeE0+uZ58/BGK0IGHByHBDiXtzMhdJr15HTYjxK7(
gK+8zp4L9IbvLGDMJ08vft32XPEWuvI8twCzFH+CsWLQADZMZKSsBas0Z/h1FwhdMgCMcY+(
JZKjTSu3i7vhvx6RzdSedXEMNTZWN4qlIx3kR5aHcukCgYA9T+Zrvm1F0seQPbLknn7EqhX]
P8TTvW/6bdPi23ExzxZn7KOdrfclYRph1LHMpAONv/x2xALIf91UB+v5ohy1oDoasL0qij1h
2ERKKdwz0ZL9SWg6VTdhr/5G994CK72fy5WhyERbDjUIdHaK3M849JJuf8cSrvSb4g==
-----END RSA PRIVATE KEY-----</keyMaterial>
</CreateKeyPairResponse>
```

# **Related Operations**

- <u>DescribeKeyPairs</u>
- <u>DeleteKeyPair</u>
- RunInstances

# CreateSecurityGroup

The CreateSecurityGroup operation creates a new security group.

Every instance is launched in a security group. If no security group is specified during launch, the instances are launched in the default security group. Instances within the same security group have unrestricted network access to each other. Instances will reject network access attempts from other instances in a different security group. As the owner of instances you can grant or revoke specific permissions using the <a href="https://doi.org/10.1001/journal.or

## **Request Parameters**

The following table describes the request parameters for CreateSecurityGroup. Parameter names are case sensitive.

Name	Description	Required
groupName	Name of the new security group.	Yes Type: xsd:string
groupDescription	Description of the new security group. Type: xsd:string	Yes

#### **Response Elements**

The following table describes the default response tags included in CreateSecurityGroup responses.

Name	Description
returnx	true if call succeeded. Type: xsd:boolean

#### **Sample Request**

#### **Sample Response**

<CreateSecurityGroupResponse xmlns="http://ec2.amazonaws.com/doc/2007-0{
 <return>true</return>
 </CreateSecurityGroupResponse>

#### **Related Operations**

- RunInstances
- <u>DescribeSecurityGroups</u>
- <u>AuthorizeSecurityGroupIngress</u>
- RevokeSecurityGroupIngress
- DeleteSecurityGroup

# DeleteKeyPair

The DeleteKeyPair operation deletes a key pair.

## **Request Parameters**

The following table describes the request parameters for DeleteKeyPair. Parameter names are case sensitive.

Name	Description	Required
keyName	Name of the key pair to delete. Type: xsd:string	Yes

#### **Response Elements**

The following table describes the default response tags included in DeleteKeyPair responses.

Name	Description
return	true if the key was successfully deleted. Type: xsd:boolean

#### **Sample Request**

# **Sample Response**

```
<DeleteKeyPair xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
    <return>true</return>
    </DeleteKeyPair>
```

# **Related Operations**

- <u>CreateKeyPair</u>
- <u>DescribeKeyPairs</u>

# **DeleteSecurityGroup**

The DeleteSecurityGroup operation deletes a security group.



If you attempt to delete a security group that contains instances, a fault is returned.

If you attempt to delete a security group that is referenced by another security group, a fault is returned. For example, if security group B has a rule that allows access from security group A, security group A cannot be deleted until the allow rule is removed.

#### **Request Parameters**

The following table describes the request parameters for DeleteSecurityGroup. Parameter names are case sensitive.

Name	Description	Required
groupName	Name of the security group to delete. Type: xsd:string	Yes

## **Response Elements**

The following table describes the default response tags included in DeleteSecurityGroup responses.

Name	Description
return	true if the group is deleted. Otherwise, false. Type: xsd:boolean

# **Sample Request**

#### **Sample Response**

<DeleteSecurityGroupResponse xmlns="http://ec2.amazonaws.com/doc/2007-08
 <return>true</return>
 </DeleteSecurityGroupResponse>

#### **Related Operations**

- <u>CreateSecurityGroup</u>
- <u>DescribeSecurityGroups</u>
- <u>AuthorizeSecurityGroupIngress</u>
- RevokeSecurityGroupIngress

# DeregisterImage

The DeregisterImage operation deregisters an AMI. Once deregistered, instances of the AMI can no longer be launched.

## **Request Parameters**

The following table describes the request parameters for DeregisterImage. Parameter names are case sensitive.

Name	Description	Required
imageId	Unique ID of the AMI which was assigned during registration (see <a href="RegisterImage">RegisterImage</a> ). For information on viewing the IDs of AMIs you own, see <a href="DescribeImages">DescribeImages</a> . Type: xsd:string	Yes

## **Response Elements**

The following table describes the default response tags included in DeregisterImage responses.

Name	Description
	true if deregistration succeeded; otherwise false. Type: xsd:boolean

#### **Sample Request**

#### **Sample Response**

<DeregisterImageResponse xmlns="http://ec2.amazonaws.com/doc/2007-08-29'
 <return>true</return>
 </DeregisterImageResponse>

# **Related Operations**

- RegisterImage
- <u>DescribeImages</u>

# DescribeImageAttribute

The DescribeImageAttribute operation returns information about an attribute of an AMI. Only one attribute can be specified per call.

## **Request Parameters**

The following table describes the request parameters for DescribeImageAttribute. Parameter names are case sensitive.

Name	Description	Required
imageId	ID of the AMI for which an attribute will be described.	Yes Type: xsd:string
launchPermission	Describes launch permissions of the AMI.  Type: ec2:EmptyElementType	Choice
productCodes	Describes the product code associated with the AMI. Type: <a href="mailto:ec2:EmptyElementType">ec2:EmptyElementType</a>	Choice

## **Response Elements**

The following table describes the default response tags included in <code>DescribeImageAttribute</code> responses.

Name	Description
imageId	ID of the AMI of which parameters are described. Type: xsd:string
launchPermission	Launch permissions of the AMI. Returned if <code>launchPermission</code> is specified. Type: <a href="mailto:ec2:LaunchPermissionItemType">ec2:LaunchPermissionItemType</a> []
productCodes	Product codes of the AMI. Returned if <i>productCodes</i> is specified.  Type: ec2:ProductCodeItemType[]

#### **Sample Request - Launch Permission**

<DescribeImageAttribute xmlns="http://ec2.amazonaws.com/doc/2007-08-29";
 <imageId>ami-61a54008</imageId>
 <launchPermission />
 </DescribeImageAttribute>

#### **Sample Response - Launch Permission**

#### **Sample Request - Product Codes**

#### **Sample Response - Product Codes**

## **Related Operations**

- <u>DescribeImages</u>
- ModifyImageAttribute
- ResetImageAttribute

## **DescribeImages**

The DescribeImages operation returns information about AMIs available to the user. This includes public AMIs available for any user to launch, private AMIs owned by the user making the request, and private AMIs owned by other users for which the user has explicit launch permissions.

Launch permissions fall into three categories:

Launch Permission	Description
public	The owner of the AMI granted launch permissions for the AMI to the all group. All users have launch permissions for these AMIs.
explicit	The owner of the AMI granted launch permissions to a specific user.
implicit	A user has implicit launch permissions for all AMIs he or she owns.

The list of AMIs returned can be modified by specifying AMI IDs, AMI owners, or users with launch permissions. If no options are specified, Amazon EC2 returns all AMIs for which the user has launch permissions.

If you specify one or more AMI IDs, only AMIs that have the specified IDs are returned. If you specify an invalid AMI ID, a fault is returned. If you specify an AMI ID for which you do not have access, it will not be included in the returned results.

If you specify one or more AMI owners, only AMIs from the specified owners and for which you have access are returned. The results can include the account IDs of the specified owners, *amazon* for AMIs owned by Amazon or *self* for AMIs that you own.

If you specify a list of executable users, only users that have launch permissions for the AMIs are returned. You can specify account IDs (if you own the AMI(s)), *self* for AMIs for which you own or have explicit permissions, or *all* for public AMIs.

Deregistered images are included in the returned results for an unspecified interval after deregistration.

### **Request Parameters**

The following table describes the request parameters for DescribeImages. Parameter names are case sensitive.

Name	Description	Required
imageSet	AMI IDs to describe Type: xsd:string[]	Yes (but can be empty)
ownersSet	Owners of AMIs to describe Type: xsd:string[]	Yes (but can be empty)
executableBySet	AMIs for which specified users have access Type: xsd:string[]	Yes (but can be empty)

### **Response Elements**

The following table describes the default response tags included in DescribeImages responses.

Name	Description
imagesSet	A list of image descriptions
	Type: <u>ec2:DescribeImagesResponseItemType[]</u>

#### **Sample Request**

#### **Sample Response**

```
<DescribeImagesResponse xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
 <imagesSet>
   <item>
     <imageId>ami-61a54008
     <imageLocation>aes-ttylinux/image.manifest.xml</imageLocation>
     <imageState>available</imageState>
     <imageOwnerId>UYY3TLBUXIEON5NQVUUX60MPWBZIQNFM</imageOwnerId>
     <isPublic>true</isPublic>
     coductCodes>
       <item>
         code>774F4FF8
       </item>
     </productCodes>
   </item>
 </imagesSet>
</DescribeImagesResponse>
```

## **Related Operations**

- <u>DescribeInstances</u>
- <u>DescribeImageAttribute</u>

### **DescribeInstances**

The DescribeInstances operation returns information about instances that you own.

If you specify one or more instance IDs, Amazon EC2 returns information for those instances. If you do not specify instance IDs, Amazon EC2 returns information for all relevant instances. If you specify an invalid instance ID, a fault is returned. If you specify an instance that you do not own, it will not be included in the returned results.

Recently terminated instances might appear in the returned results. This interval is usually less than one hour.

### **Request Parameters**

The following table describes the request parameters for DescribeInstances. Parameter names are case sensitive.

Name	Description	Required
instancesSet	Set of instances IDs to get the status of. Type: xsd:string[]	Yes (but can be empty)

### **Response Elements**

The following table describes the default response tags included in <code>DescribeInstances</code> responses.

Name	Description
reservationSet	A list of structures describing the status of all requested instances. Type: <a href="mailto:ec2:ReservationInfoType">ec2:ReservationInfoType</a> []

### **Sample Request**

#### **Sample Response**

```
<DescribeInstancesResponse xmlns="http://ec2.amazonaws.com/doc/2007-08-2</pre>
  <reservationSet>
    <item>
      <reservationId>r-44a5402d</reservationId>
      <ownerId>UYY3TLBUXIEON5NQVUUX60MPWBZIQNFM</ownerId>
      <qroupSet>
        <item>
          <groupId>default
        </item>
     </groupSet>
      <instancesSet>
        <item>
          <instanceId>i-28a64341</instanceId>
         <imageId>ami-6ea54007/imageId>
         <instanceState>
           <code>0</code>
           <name>running</name>
         </instanceState>
          <privateDnsName>domU-12-31-35-00-1E-01.compute-1.internal
          <dnsName>ec2-72-44-33-4.compute-1.amazonaws.com</dnsName>
          <keyName>example-key-name</keyName>
         <amiLaunchIndex>23</amiLaunchIndex>
          cproductCodesSet>
            <item><productCode>774F4FF8</productCode></item>
          <instanceType>m1.large</instanceType>
          <launchTime>2007-08-07T11:54:42.000Z</launchTime>
        </item>
      </instancesSet>
    </item>
 </reservationSet>
</DescribeInstancesResponse>
```

## **Related Operations**

- RunInstances
- <u>TerminateInstances</u>

## **DescribeKeyPairs**

The DescribeKeyPairs operation returns information about key pairs available to you. If you specify key pairs, information about those key pairs is returned. Otherwise, information for all registered key pairs is returned.

### **Request Parameters**

The following table describes the request parameters for DescribeKeyPairs. Parameter names are case sensitive.

Name	Description	Required
keySet	Key pair IDs to describe.  Type: xsd:string[]	Yes (but can be empty)

### **Response Elements**

The following table describes the default response tags included in DescribeKeyPairs responses.

Name	Description
keySet	A list of key pair descriptions
	Type: <u>ec2:DescribeKeypairsResponseItemType[]</u>

### **Sample Request**

#### **Sample Response**

## **Related Operations**

- <u>CreateKeypair</u>
- <u>DeleteKeypair</u>
- RunInstances

## DescribeSecurityGroups

The DescribeSecurityGroups operation returns information about security groups that you own.

If you specify security group names, information about those security group is returned. Otherwise, information for all security group is returned. If you specify a group that does not exist, a fault is returned.

### **Request Parameters**

The following table describes the request parameters for DescribeSecurityGroups. Parameter names are case sensitive.

Name	Description	Required
securityGroupSet	List of security groups to describe. Type: xsd:string[]	Yes

### **Response Elements**

The following table describes the default response tags included in DescribeSecurityGroups responses.

Name	Description
securityGroupInfo	Information about security groups.  Type: ec2:SecurityGroupItemType[]

#### **Sample Request**

#### **Sample Response**

```
<DescribeSecurityGroupsResponse xmlns="http://ec2.amazonaws.com/doc/2007</pre>
  <securityGroupInfo>
    <item>
      <ownerId>UYY3TLBUXIEON5NQVUUX60MPWBZIQNFM</ownerId>
      <groupName>WebServers
      <groupDescription>Web</groupDescription>
      <ipPermissions>
        <item>
         <ipProtocol>tcp</ipProtocol>
          <fromPort>80</fromPort>
          <toPort>80</toPort>
          <qroups/>
         <ipRanges>
            <item>
              <cidrIp>0.0.0.0/0</cidrIp>
            </item>
          </ipRanges>
         </item>
      </ipPermissions>
    </item>
    <item>
      <ownerId>UYY3TLBUXIEON5NQVUUX60MPWBZIQNFM</ownerId>
     <groupName>RangedPortsBySource
     <groupDescription>A</groupDescription>
      <ipPermissions>
        <item>
          <ipProtocol>tcp</ipProtocol>
         <fromPort>6000</fromPort>
          <toPort>7000</toPort>
          <qroups/>
          <ipRanges/>
        </item>
     </ip>
    </item>
 </securityGroupInfo>
</DescribeSecurityGroupsResponse>
```

### **Related Operations**

- <u>CreateSecurityGroup</u>
- <u>AuthorizeSecurityGroupIngress</u>
- RevokeSecurityGroupIngress
- DeleteSecurityGroup

## GetConsoleOutput

The GetConsoleOutput operation retrieves console output for the specified instance.

Instance console output is buffered and posted shortly after instance boot, reboot, and termination. Amazon EC2 preserves the most recent 64 KB output which will be available for at least one hour after the most recent post.

### **Request Parameters**

The following table describes the request parameters for GetConsoleOutput. Parameter names are case sensitive.

Name	Description	Required
InstanceId	An instance ID returned from a previous call to RunInstances. Type: xsd:string	Yes

### **Response Elements**

The following table describes the default response tags included in GetConsoleOutput responses.

Name	Description
instanceId	The instance ID. Type: xsd:string
timestamp	The time the output was last updated. Type: xsd:dateTime
output	The console output, Base64 encoded. Type: xsd:string

## **Sample Request**

#### **Sample Response**

# ${\bf Modify Image Attribute}$

The ModifyImageAttribute operation modifies an attribute of an AMI.

### **Request Parameters**

The following table describes the request parameters for ModifyImageAttribute. Parameter names are case sensitive.

Name	Description	Required
imageId	AMI ID to modify. Type: xsd:string	Yes
launchPermission	Adds or removes launch permissions for the AMI.  Type: ec2:LaunchPermissionOperationType	Choice
productCodes	Attaches a product code to the AMI, allowing developers to charge for the use of their AMIs. Currently only one product code can be associated with an AMI. Once set, the product code can not be changed or reset.  Type: ec2:ProductCodeItemType[]	Choice

### **Response Elements**

The following table describes the default response tags included in ModifyImageAttribute responses.

Name	Description
	true if the operation succeeded, otherwise false. Type: xsd:boolean

#### **Sample Request - Launch Permission**

#### **Sample Request - Product Codes**

## **Sample Response**

# **Related Operations**

- ResetImageAttribute
- <u>DescribeImageAttribute</u>

## RebootInstances

The RebootInstances operation requests a reboot of one or more instances. This operation is asynchronous; it only queues a request to reboot the specified instance(s). The operation will succeed if the instances are valid and belong to the user. Requests to reboot terminated instances are ignored.

## **Request Parameters**

The following table describes the request parameters for RebootInstances. Parameter names are case sensitive.

Name	Description	Required
instancesSet	One or more instance IDs. Type: xsd:string[]	Yes

## **Response Elements**

The following table describes the default response tags included in RebootInstances responses.

Name	Description
result	true if the operation succeeded. Type: xsd:boolean

## **Sample Request**

## **Sample Response**

# RegisterImage

The RegisterImage operation registers an AMI with Amazon EC2. Images must be registered before they can be <u>launched</u>.

Each AMI is associated with an unique ID which is provided by the Amazon EC2 service through the RegisterImage operation. During registration, Amazon EC2 retrieves the specified image manifest from Amazon S3 and verifies that the image is owned by the user registering the image.

The image manifest is retrieved once and stored within the Amazon EC2. Any modifications to an image in Amazon S3 invalidates this registration. If you make changes to an image, <u>deregister</u> the previous image and register the new image.

## **Request Parameters**

The following table describes the request parameters for RegisterImage. Parameter names are case sensitive.

Name	Description	Required
	Full path to your AMI manifest in Amazon S3 storage. Type: xsd:string	Yes

## **Response Elements**

The following table describes the default response tags included in RegisterImage responses.

Name	Description
imageId	Unique ID of the newly registered machine image. Type: xsd:string

## **Sample Request**

## **Sample Response**

# **Related Operations**

- <u>DescribeImages</u>
- <u>DeregisterImage</u>

# ResetImageAttribute

The ResetImageAttribute operation resets an attribute of an AMI to its default value.



The productCodes attribute cannot be reset.

## **Request Parameters**

The following table describes the request parameters for ResetImageAttribute. Parameter names are case sensitive.

Name	Description	Required
imageId	ID of the AMI on which the attribute will be reset. Type: xsd:string	Yes
launchPermission	Resets the AMI's launch permissions. All public and explicit launch permissions for the AMI are revoked.  Type: ec2:EmptyElementType	Yes

## **Response Elements**

The following table describes the default response tags included in ResetImageAttribute responses.

Name	Description
return	true if the operation succeeded, otherwise false. Type: xsd:boolean

## **Sample Request**

```
<ResetImageAttribute xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
    <imageId>ami-61a54008</imageId>
        <launchPermission />
        </ResetImageAttribute>
```

## **Sample Response**

<ResetImageAttributeResponse xmlns="http://ec2.amazonaws.com/doc/2007-08
<return>true</return>

</ResetImageAttributeResponse>

## **Related Operations**

- ModifyImageAttribute
- <u>DescribeImageAttribute</u>

# RevokeSecurityGroupIngress

The RevokeSecurityGroupIngress operation revokes permissions from a security group. The permissions used to revoke must be specified using the same values used to grant the permissions.

Permissions are specified by IP protocol (TCP, UDP, or ICMP), the source of the request (by IP range or an Amazon EC2 user-group pair), the source and destination port ranges (for TCP and UDP), and the ICMP codes and types (for ICMP).

Permission changes are quickly propagated to instances within the security group. However, depending on the number of instances in the group, a small delay is might occur, .

## **Request Parameters**

The following table describes the request parameters for RevokeSecurityGroupIngress. Parameter names are case sensitive.

Name	Description	Required
userId	AWS Access Key ID. Type: xsd:string	Yes
groupName	Name of the group to modify. Type: xsd:string	Yes
ipPermissions	Set of permissions to remove from the group. Type: ec2:IpPermissionType[]	Yes

## **Response Elements**

The following table describes the default response tags included in RevokeSecurityGroupIngress responses.

Name	Description
return	true if permissions successfully revoked. Type: xsd:boolean

#### **Sample Request**

## **Sample Response**

<RevokeSecurityGroupIngressResponse xmlns="http://ec2.amazonaws.com/doc/
 <return>true</return>
 </RevokeSecurityGroupIngressResponse>

## **Related Operations**

- <u>CreateSecurityGroup</u>
- <u>DescribeSecurityGroups</u>
- <u>AuthorizeSecurityGroupIngress</u>
- DeleteSecurityGroup

#### **RunInstances**

The RunInstances operation launches a specified number of instances.

If Amazon EC2 cannot launch the minimum number AMIs you request, no instances will be launched. If there is insufficient capacity to launch the maximum number of AMIs you request, Amazon EC2 launches the minimum number specified for each AMI and allocate the remaining available instances using round robin.

In the following example, Libby generates a request to launch two images (database and web\_server):

- 1. Libby runs the RunInstances operation to launch database instances (min. 10, max. 15) and web\_server instances (min. 30, max. 40).
  - Because there are currently 30 instances available and Libby needs a minimum of 40, no instances are launched.
- 2. Libby adjusts the number of instances she needs and runs the RunInstances operation to launch database instances (min. 5, max. 10) and web\_server instances (min. 20, max. 40).
  - Amazon EC2 launches the minimum number of instances for each AMI (5 database, 20 web\_server).
  - The remaining 5 instances are allocated using round robin.
- 3. Libby adjusts the number of instances she needs and runs the RunInstances operation again to launch database instances (min. 5, max. 10) and web\_server instances (min. 20, max. 40).



Every instance is launched in a <u>security group</u>. If you do not specify a security group at launch, the instances start in your default security group.

You can provide an optional <u>key pair ID</u> for each image in the launch request. All instances that are created from images that use this key pair will have access

to the associated public key at boot. You can use this key to provide secure access to an instance of an image on a per-instance basis. Amazon EC2 public images use this feature to provide secure access without passwords.

#### **Important**

Launching public images without a key pair ID will leave them inaccessible.

The public key material is made available to the instance at boot time by placing it in the openssh\_id.pub file on a logical device that is exposed to the instance as /dev/sda2 (the ephemeral store). The format of this file is suitable for use as an entry within ~/.ssh/authorized\_keys (the OpenSSH format). This can be done at boot (e.g., as part of rc.local) allowing for secure access without passwords.

Optional user data can be provided in the launch request. All instances comprising the launch request have access to this data (see Instance Metadata for details).



If any of the AMIs have a product code attached for which the user has not subscribed, the RunInstances call will fail.

## **Request Parameters**

The following table describes the request parameters for RunInstances. Parameter names are case sensitive.

Name	Description	Required
instancesSet	Description of the instances to launch.  Type: ec2:RunInstanceItemType[]	Yes
groupSet	Description of the security groups with which to associate the instances. Type: <a href="ec2:GroupSetType">ec2:GroupSetType</a> []	Yes
userData	The user data available to the launched instances.  Type: ec2:UserDataType	No
addressingType	The addressing scheme with which to launch the instance. The supported addressing type is public. For the public scheme, the instance has a private and public IP address that are mapped through NAT. For more information, see <a href="Instance Addressing">Instance Addressing</a> .  Note  To support our legacy cluster, early beta customers can also specify direct. In the direct scheme, the instance has a single public IP address.	No
	Type: xsd:string	
instanceType	This specifies the instance type.  Options include m1.small, m1.large, and m1.xlarge.  The default value is m1.small.  For more information on instance types, see Selecting Instance Types  Type: xsd:string	No

## **Response Elements**

The following table describes the default response tags included in RunInstances responses.

Name	Description
RunInstancesResponse	Status information about the instances launched. Type: <a href="mailto:ec2:ReservationInfoType">ec2:ReservationInfoType</a>

#### **Sample Request**

#### **Sample Response**

```
<RunInstancesResponse xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
 <reservationId>r-47a5402e</reservationId>
 <ownerId>495219933132/ownerId>
 <qroupSet>
    <item>
      <groupId>default
    </item>
 </groupSet>
 <instancesSet>
    <item>
      <instanceId>i-2ba64342</instanceId>
      <imageId>ami-60a54009</imageId>
     <instanceState>
        <code>0</code>
        <name>pending</name>
      </instanceState>
      <privateDnsName></privateDnsName>
      <dnsName></dnsName>
      <keyName>example-key-name</keyName>
      <amiLaunchIndex>0</amiLaunchIndex>
     <instanceType>m1.small</instanceType>
      <launchTime>2007-08-07T11:51:50.000Z</launchTime>
    </item>
    <item>
      <instanceId>i-2bc64242</instanceId>
      <imageId>ami-60a54009/imageId>
      <instanceState>
        <code>0</code>
        <name>pending</name>
      </instanceState>
      <privateDnsName></privateDnsName>
      <dnsName></dnsName>
     <keyName>example-key-name</keyName>
      <amiLaunchIndex>1</amiLaunchIndex>
      <instanceType>m1.small</instanceType>
      <launchTime>2007-08-07T11:51:50.000Z</launchTime>
    </item>
    <item>
      <instanceId>i-2be64332</instanceId>
      <imageId>ami-60a54009</imageId>
      <instanceState>
        <code>0</code>
        <name>pending</name>
      </instanceState>
```

## **Related Operations**

- <u>DescribeInstances</u>
- <u>TerminateInstances</u>
- <u>AuthorizeSecurityGroupIngress</u>
- RevokeSecurityGroupIngress
- <u>DescribeSecurityGroups</u>

## **TerminateInstances**

The TerminateInstances operation shuts down one or more instances. This operation is idempotent; if you terminate an instance more than once, each call will succeed.

Terminated instances will remain visible after termination (approximately one hour).

## **Request Parameters**

The following table describes the request parameters for TerminateInstances. Parameter names are case sensitive.

Name	Description	Required
instancesSet	One or more instance IDs. Type: xsd:string[]	Yes

## **Response Elements**

The following table describes the default response tags included in TerminateInstances responses.

Name	Description
instancesSet	A complex type describing the current and new state of each instance specified. Type: <a href="ec2:TerminateInstancesResponseInfoType">ec2:TerminateInstancesResponseInfoType</a> []

#### **Sample Request**

#### **Sample Response**

# **Related Operations**

• <u>DescribeInstances</u>

# **Amazon EC2 Query API**

The Amazon EC2 API consists of web service operations for every task the service can perform. This section describes each operation in detail.

# **Common Query Parameters**

#### **Request Parameters**

All Query operations share a set of common parameters that must be present in each call:

Name	Description	Required
Action	Indicates the action to perform. Example: RunInstances	Yes
Version	The API version to use, as specified in the WSDL. Example: 2007-08-29	Yes
AWSAccessKeyId	The Access Key ID for the request sender. This identifies the account which will be charged for usage of the service. The account with which the Access Key ID is associated must be signed up for Amazon EC2, or requests will not be accepted. 10QMXFEV71ZS32XQFTR2	Yes
Timestamp	The date and time at which the request is signed, in the format YYYY-MM-DDThh:mm:ssZ, as specified in the <a href="ISO 8601">ISO 8601</a> standard. Example: 2006-07-07T15:04:56Z	Yes
Expires	The date and time at which the signature included in the request expires, in the format YYYY-MM-DDThh:mm:ssZ, as specified in the ISO 8601 standard.  Example: 2006-07-07T15:04:56Z	Yes
Signature	A request signature is calculated as explained in Request Authentication.  Example: Qnp14Qk/7tINHzfXCiT7VbBatDA=	Yes
SignatureVersion	A value of 0 or 1 indicates the method chosen to construct the string to be signed. Currently, only a value of 1 is valid. Example: 1	Yes



The *Timestamp* parameter can be used instead of *Expires*. Requests must include either *Timestamp* or *Expires*, but cannot contain both.

Parameter values must be URL-encoded. This is true for any Query parameter passed to Amazon EC2 and is typically necessary in the *Signature* parameter. Some clients do this automatically, but this is not the norm.

# **List of Operations by Function**

#### **Images**

- RegisterImage
- DescribeImages
- <u>DeregisterImage</u>

#### **Instances**

- RunInstances
- <u>DescribeInstances</u>
- <u>TerminateInstances</u>
- ConfirmProductInstance

#### **Key Pairs**

- <u>CreateKeyPair</u>
- <u>DescribeKeyPairs</u>
- <u>DeleteKeyPair</u>

#### **Image Attributes**

- ModifyImageAttribute
- <u>DescribeImageAttribute</u>
- ResetImageAttribute

#### **Security Groups**

• CreateSecurityGroup

- <u>DescribeSecurityGroups</u>
- <u>DeleteSecurityGroup</u>
- <u>AuthorizeSecurityGroupIngress</u>
- RevokeSecurityGroupIngress

## AuthorizeSecurityGroupIngress

The AuthorizeSecurityGroupIngress operation adds permissions to a security group.

Permissions are specified by the IP protocol (TCP, UDP or ICMP), the source of the request (by IP range or an Amazon EC2 user-group pair), the source and destination port ranges (for TCP and UDP), and the ICMP codes and types (for ICMP). When authorizing ICMP, -1 can be used as a wildcard in the type and code fields.

Permission changes are propagated to instances within the security group as quickly as possible. However, depending on the number of instances, a small delay might occur.

When authorizing a user/group pair permission, <code>GroupName</code>, <code>SourceSecurityGroupName</code> and <code>SourceSecurityGroupOwnerId</code> must be specified. When authorizing a CIDR IP permission, <code>GroupName</code>, <code>IpProtocol</code>, <code>FromPort</code>, <code>ToPort</code> and <code>CidrIp</code> must be specified. Mixing these two types of parameters is not allowed.

### **Request Parameters**

The following table describes the request parameters for AuthorizeSecurityGroupIngress. Parameter names are case sensitive.

Name	Description	Required
GroupName	Name of the group to modify. Type: String	Yes
SourceSecurityGroupName	Name of security group to authorize access to when operating on a user/group pair.  Type: String	When authorizing user/group pair permission.
SourceSecurityGroupOwnerId	Owner of security group to authorize access to when operating on a user/group pair.  Type: String	When authorizing user/group pair permisison.
IpProtocol	IP protocol to authorize access to when operating on a CIDR IP.  Type: String  Valid Values: tcp   udp   icmp	When authorizing CIDR IP permission.
FromPort	Bottom of port range to authorize access to when operating on a CIDR IP. This contains the ICMP type if ICMP is being authorized.  Type: Int	When authorizing CIDR IP permission.
ToPort	Top of port range to authorize access to when operating on a CIDR IP. This contains the ICMP code if ICMP is being authorized.  Type: Int	When authorizing CIDR IP permission.
CidrIp	CIDR IP range to authorize access to when operating on a CIDR IP. Type: String	When authorizing CIDR IP permission.

#### **Response Elements**

The following table describes the default response tags included in AuthorizeSecurityGroupIngress responses.

Name	Description
return	true if permissions successfully added. Type: xsd:boolean

#### **Sample Request**

```
https://ec2.amazonaws.com/
?Action=AuthorizeSecurityGroupIngress
&IpProtocol=tcp
&FromPort=80
&ToPort=80
&CidrIp=0.0.0.0/0
&...auth parameters...
```

#### **Sample Response**

<AuthorizeSecurityGroupIngressResponse xmlns="http://ec2.amazonaws.com/c
<return>true</return>
</AuthorizeSecurityGroupIngressResponse>

### **Related Operations**

- <u>CreateSecurityGroup</u>
- <u>DescribeSecurityGroups</u>
- RevokeSecurityGroupIngress
- DeleteSecurityGroup

#### **ConfirmProductInstance**

The ConfirmProductInstance operation returns true if the specified product code is attached to the specified instance. The operation returns false if the product code is not attached to the instance.

The ConfirmProductInstance operation can only be executed by the owner of the AMI. This feature is useful when an AMI owner is providing support and wants to verify whether a user's instance is eligible.

### **Request Parameters**

The following table describes the request parameters for ConfirmProductInstance. Parameter names are case-sensitive.

Name	Description	Required
ProductCode	The product code to confirm. Type: String	Yes
InstanceId	The instance for which to confirm the product code. Type: String	Yes

### **Response Elements**

The following table describes the default response tags included in ConfirmProductInstance responses.

Name	Description
result	true if the product code is attached to the instance, false if it is not. Type: xsd:boolean
ownerId	The instance owner's account ID. Only present if the product code is attached to the instance.  Type: xsd:string

#### **Sample Request**

https://ec2.amazonaws.com/ ?Action=ConfirmProductInstance &ProductCode=774F4FF8 &InstanceId=i-10a64379 &...auth parameters...

#### **Sample Response**

```
<ConfirmProductInstanceResponse xmlns="http://ec2.amazonaws.com/doc/2007
    <result>true</result>
        <ownerId>254933287430</ownerId>
        </ConfirmProductInstanceResponse>
```

# **Related Operations**

- <u>DescribeInstances</u>
- RunInstances

# CreateKeyPair

The CreateKeyPair operation creates a new 2048 bit RSA key pair and returns a unique ID that can be used to reference this key pair when <u>launching</u> new instances.

#### **Request Parameters**

The following table describes the request parameters for CreateKeyPair. Parameter names are case sensitive.

Name	Description	Required
KeyName	A unique name for the key pair. Type: String	Yes

### **Response Elements**

The following table describes the default response tags included in CreateKeyPair responses.

Name	Description
keyName	The key pair name provided in the original request. Type: xsd:string
KeyFingerprint	A SHA-1 digest of the DER encoded private key. Type: xsd:string
KeyMaterial	An unencrypted PEM encoded RSA private key. Type: xsd:string

#### **Sample Request**

https://ec2.amazonaws.com/ ?Action=CreateKeyPair &KeyName=example-key-name &...auth parameters...

#### **Sample Response**

```
<CreateKeyPairResponse xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
  <keyName>example-key-name</keyName>
  <keyFingerprint>1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:
  <keyMaterial>----BEGIN RSA PRIVATE KEY-----
MIIEoQIBAAKCAQBuLFq5ujHrtmljnutSuo08Xe56LlT+HM8v/xkaa39EstM3/aFxTHqElQi.
HungXQ29VTc8rc1bW0lkdi230H5eqkMHGhvEwqa0HWASUMll4o3o/IX+0f2UcPoKC0VUR+j>
5AU52EOfanIn3Z08lFW7Edp5a3q4DhjGlUKToHVbicL5E+q45zfB95wIyywWZfeW/UUF3Lp(
ebIUlq1qTbHkLbCC2r7RTn8vpQWp47BGVYGtGSBMpTRP5hnbzzuqj3itkiLHjU39S2sJCJ07
i8BygR4s3mHKBj8l+ePQxG1kGbF6R4yg6sECmXn17MRQVX0DNHZbAgMBAAECggEAY1tsiUs]
91CXirkYGuVfLvLflXenxfI50mDFms/mumTqloH07tr0oriHDR5K7wMcY/YY5YkcXNo7mvU\
ZNUJs7rw9gZRTrf7LylaJ58k0cyajw8TsC4e4LPbFaHwS1d6K8rXh64o6WgW4SrsB6ICmr1
3wcfqt5ecIu4TZf00E9IHjn+2eRlsrjBde0Ri7KiUNC/pAG23I6MdD0FEQRcCSigCj+4/mci
SWS4dMbrpb9FNSIcf9dcLxVM7/6KxqJNfZc9XWzUw77Jq8x92Zd0fVhH0ux5IZC+UvSKWB4c
tE8C3p9bbU9VGyY5vLCAiIb4qQKBqQDLi024GXrIkswF32YtBBMuVqLGCwU9h9Hl09mKAc2r
jUE5IpzRjTedc9I2qiIMUTwtgnw42auSCzbUeYMURPtDqyQ7p6AjMujp9EPemcSV0K9vXYL(
xW9MC0dtV6iPkCN7q0qiZXPRKaFbWADp16p8UAIvS/a5XXk5jwKBqQCKkpHi2EISh1uRkhxl
iDCiK6JBRsMvpLbc0v5dKwP5alo1fmdR5PJaV2qvZSj5CYNpMAy1/EDNTY50SIJU+0KFmQby
rdLNLDL4+TcnT7c62/aH01ohYaf/VCbRhtLlBfqGo0c7+sAc8vmKkesnF7CqCEKDyF/dhrx\
qC0iZzzNAapayz1+JcVTwwEid6j9JqNXbBc+Z2YwMi+T0Fv/P/hwkX/ype0XnIUcw0Ih/Yt(
DQbsz7LcY1HqXiHKYNWNvXgww0+oiChjxvEkSdsTTIfnK4VSCvU9BxDbQHjdiNDJbL6oar91
rBYvChJZF7LvUH4YmVpHAoGAbZ2X7XvoeE0+uZ58/BGK0IGHByHBDiXtzMhdJr15HTYjxK7(
gK+8zp4L9IbvLGDMJ08vft32XPEWuvI8twCzFH+CsWLQADZMZKSsBas0Z/h1FwhdMgCMcY+(
JZKjTSu3i7vhvx6RzdSedXEMNTZWN4qlIx3kR5aHcukCgYA9T+Zrvm1F0seQPbLknn7EqhX]
P8TTvW/6bdPi23ExzxZn7KOdrfclYRph1LHMpAONv/x2xALIf91UB+v5ohy1oDoasL0qij1h
2ERKKdwz0ZL9SWg6VTdhr/5G994CK72fy5WhyERbDjUIdHaK3M849JJuf8cSrvSb4g==
-----END RSA PRIVATE KEY-----</keyMaterial>
</CreateKeyPairResponse>
```

## **Related Operations**

- <u>DescribeKeyPairs</u>
- <u>DeleteKeyPair</u>
- RunInstances

# CreateSecurityGroup

The CreateSecurityGroup operation creates a new security group.

Every instance is launched in a security group. If no security group is specified during launch, the instances are launched in the default security group. Instances within the same security group have unrestricted network access to each other. Instances will reject network access attempts from other instances in a different security group. As the owner of instances you can grant or revoke specific permissions using the <a href="mailto:AuthorizeSecurityGroupIngress">AuthorizeSecurityGroupIngress</a> and <a href="mailto:RevokeSecurityGroupIngress">RevokeSecurityGroupIngress</a> operations.

### **Request Parameters**

The following table describes the request parameters for CreateSecurityGroup. Parameter names are case sensitive.

Name	Description	Required
GroupName	Name of the new security group. Type: String	Yes
GroupDescription	Description of the new security group. Type: String	Yes

#### **Response Elements**

The following table describes the default response tags included in CreateSecurityGroup responses.

Name	Description
return	true if call succeeded.
	Type: xsd:boolean

#### **Sample Request**

https://ec2.amazonaws.com/ ?Action==CreateSecurityGroup &GroupName=WebServers &GroupDescription=Web &...auth parameters...

#### **Sample Response**

<CreateSecurityGroupResponse xmlns="http://ec2.amazonaws.com/doc/2007-0{
 <return>true</return>
 </CreateSecurityGroupResponse>

#### **Related Operations**

- RunInstances
- <u>DescribeSecurityGroups</u>
- <u>AuthorizeSecurityGroupIngress</u>
- RevokeSecurityGroupIngress
- DeleteSecurityGroup

# DeleteKeyPair

The DeleteKeyPair operation deletes a key pair.

### **Request Parameters**

The following table describes the request parameters for DeleteKeyPair. Parameter names are case sensitive.

Name	Description	Required
KeyName	Name of the key pair to delete. Type: String	Yes

#### **Response Elements**

The following table describes the default response tags included in DeleteKeyPair responses.

Name	Description
return	true if the key was successfully deleted. Type: xsd:boolean

#### **Sample Request**

https://ec2.amazonaws.com/ ?Action=DeleteKeyPair &KeyName=example-key-name &...auth parameters...

## **Sample Response**

```
<DeleteKeyPair xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
    <return>true</return>
    </DeleteKeyPair>
```

## **Related Operations**

- <u>CreateKeyPair</u>
- <u>DescribeKeyPairs</u>

# **DeleteSecurityGroup**

The DeleteSecurityGroup operation deletes a security group.



If you attempt to delete a security group that contains instances, a fault is returned.

If you attempt to delete a security group that is referenced by another security group, a fault is returned. For example, if security group B has a rule that allows access from security group A, security group A cannot be deleted until the allow rule is removed.

#### **Request Parameters**

The following table describes the request parameters for DeleteSecurityGroup. Parameter names are case sensitive.



A security group cannot be deleted if it is referenced by another security group. For example, if security group B has a rule that allows access from security group A, security group A cannot be deleted until the allow rule is removed.

Name	Description	Required
GroupName	Name of the security group to delete. Type: String	Yes

## **Response Elements**

The following table describes the default response tags included in DeleteSecurityGroup responses.

Name	Description
return	true if the group is deleted. Otherwise, false. Type: xsd:boolean

### **Sample Request**

https://ec2.amazonaws.com/ ?Action=DeleteSecurityGroup &GroupName=RangedPortsBySource &...auth parameters...

### **Sample Response**

<DeleteSecurityGroupResponse xmlns="http://ec2.amazonaws.com/doc/2007-08
 <return>true</return>
 </DeleteSecurityGroupResponse>

## **Related Operations**

- <u>CreateSecurityGroup</u>
- <u>DescribeSecurityGroups</u>
- <u>AuthorizeSecurityGroupIngress</u>
- RevokeSecurityGroupIngress

# DeregisterImage

The DeregisterImage operation deregisters an AMI. Once deregistered, instances of the AMI can no longer be launched.

## **Request Parameters**

The following table describes the request parameters for DeregisterImage. Parameter names are case sensitive.

Name	Description	Required
ImageId	Unique ID of a machine image, returned by a call to <a href="RegisterImage">RegisterImage</a> or <a href="DescribeImages">DescribeImages</a> . Type: String	Yes

## **Response Tags**

The following table describes the default response tags included in DeregisterImage responses.

Name	Description
	true if deregistration succeeded; otherwise false. Type: xsd:boolean

## **Sample Request**

https://ec2.amazonaws.com/ ?Action=DeregisterImage &ImageId=ami-61a54008 &...auth parameters...

## **Sample Response**

<DeregisterImageResponse xmlns="http://ec2.amazonaws.com/doc/2007-08-29'
 <return>true</return>
 </DeregisterImageResponse>

## **Related Operations**

- RegisterImage
- <u>DescribeImages</u>

# DescribeImageAttribute

The DescribeImageAttribute operation returns information about an attribute of an AMI. Only one attribute can be specified per call.

## **Request Parameters**

The following table describes the request parameters for DescribeImageAttribute. Parameter names are case-sensitive.

Name	Description	Required
ImageIdParam;	ID of the AMI for which an attribute will be described. Type: String	Yes
Attribute	Specifies the attribute to describe.  Type: String Valid Value: launchPermission	Yes

#### **Attributes**

Attribute Name	Description
launchPermission	The AMIs launch permissions.
productCodes	The product code attached to the AMI.

## **Response Elements**

The following table describes the default response tags included in <code>DescribeImageAttribute</code> responses.

Name	Description
imageId	ID of the AMI described. Type: xsd:string
launchPermission	Launch permissions of the AMI. Returned if <code>launchPermission</code> is specified. Type: <a href="mailto:ec2:LaunchPermissionItemType">ec2:LaunchPermissionItemType</a> []
productCodes	Product codes of the AMI. Returned if <i>productCodes</i> is specified.  Type: ec2:ProductCodeItemType[]

#### **Sample Request - Launch Permission**

https://ec2.amazonaws.com/ ?Action=DescribeImageAttribute &ImageId=ami-61a54008 &Attribute=launchPermission &...auth parameters...

#### **Sample Response - Launch Permission**

### **Sample Request - Product Codes**

https://ec2.amazonaws.com/ ?Action=DescribeImageAttribute &ImageId=ami-61a54008 &Attribute=productCodes &...auth parameters...

#### **Sample Response - Product Codes**

## **Related Operations**

- <u>DescribeImages</u>
- ModifyImageAttribute
- ResetImageAttribute

## **DescribeImages**

The DescribeImages operation returns information about AMIs available to the user. This includes public AMIs available for any user to launch, private AMIs owned by the user making the request, and private AMIs owned by other users for which the user has explicit launch permissions.

Launch permissions fall into three categories:

Launch Permission	Description
public	The owner of the AMI granted launch permissions for the AMI to the all group. All users have launch permissions for these AMIs.
explicit	The owner of the AMI granted launch permissions to a specific user.
implicit	A user has implicit launch permissions for all AMIs he or she owns.

The list of AMIs returned can be modified by specifying AMI IDs, AMI owners, or users with launch permissions. If no options are specified, Amazon EC2 returns all AMIs for which the user has launch permissions.

If you specify one or more AMI IDs, only AMIs that have the specified IDs are returned. If you specify an invalid AMI ID, a fault is returned. If you specify an AMI ID for which you do not have access, it will not be included in the returned results.

If you specify one or more AMI owners, only AMIs from the specified owners and for which you have access are returned. The results can include the account IDs of the specified owners, *amazon* for AMIs owned by Amazon or *self* for AMIs that you own.

If you specify a list of executable users, only users that have launch permissions for the AMIs are returned. You can specify account IDs (if you own the AMI(s)), *self* for AMIs for which you own or have explicit permissions, or *all* for public AMIs.

Deregistered images are included in the returned results for an unspecified interval after deregistration.

## **Request Parameters**

The following table describes the request parameters for DescribeImages. Parameter names are case sensitive.

Name	Description	Required
ImageId.n	A list of image descriptions Type: String	No
Owner. <mark>n</mark>	Owners of AMIs to describe. Type: String	No
ExecutableBy.n	AMIs for which specified users have access. Type: String	No

## **Response Elements**

The following table describes the default response tags included in DescribeImages responses.

Name	Description
imagesSet	A list of image descriptions.  Type: ec2:DescribeImagesResponseItemType[]

## **Sample Request**

https://ec2.amazonaws.com/ ?Action=DescribeImages &ImageId.1=ami-61a54008 &...auth parameters...

#### **Sample Response**

```
<DescribeImagesResponse xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
 <imagesSet>
   <item>
     <imageId>ami-61a54008
     <imageLocation>aes-ttylinux/image.manifest.xml</imageLocation>
     <imageState>available</imageState>
     <imageOwnerId>UYY3TLBUXIEON5NQVUUX60MPWBZIQNFM</imageOwnerId>
     <isPublic>false</isPublic>
     coductCodes>
       <item>
         code>774F4FF8
       </item>
     </productCodes>
   </item>
 </imagesSet>
</DescribeImagesResponse>
```

## **Related Operations**

- <u>DescribeInstances</u>
- <u>DescribeImageAttribute</u>

### **DescribeInstances**

The DescribeInstances operation returns information about instances that you own.

If you specify one or more instance IDs, Amazon EC2 returns information for those instances. If you do not specify instance IDs, Amazon EC2 returns information for all relevant instances. If you specify an invalid instance ID, a fault is returned. If you specify an instance that you do not own, it will not be included in the returned results.

Recently terminated instances might appear in the returned results. This interval is usually less than one hour.

## **Request Parameters**

The following table describes the request parameters for DescribeInstances. Parameter names are case sensitive.

Name	Description	Required
InstanceId.n	Set of instances IDs to get the status of. Type: String	No

## **Response Elements**

The following table describes the default response tags included in <code>DescribeInstances</code> responses.

Name	Description
reservationSet	A list of structures describing the status of all requested instances. Type: <a href="mailto:ec2:ReservationInfoType">ec2:ReservationInfoType</a>

## **Sample Request**

https://ec2.amazonaws.com/ ?Action=DescribeInstances &InstanceId.1=i-28a64341 &...auth parameters...

#### **Sample Response**

```
<DescribeInstancesResponse xmlns="http://ec2.amazonaws.com/doc/2007-08-2</pre>
  <reservationSet>
    <item>
      <reservationId>r-44a5402d</reservationId>
      <ownerId>UYY3TLBUXIEON5NQVUUX60MPWBZIQNFM</ownerId>
     <groupSet>
       <item>
         <groupId>default
        </item>
     </groupSet>
      <instancesSet>
        <item>
         <instanceId>i-28a64341</instanceId>
         <imageId>ami-6ea54007/imageId>
         <instanceState>
           <code>0</code>
           <name>running</name>
         </instanceState>
         <privateDnsName>domU-12-31-35-00-1E-01.compute-1.internal
         <dnsName>ec2-72-44-33-4.compute-1.amazonaws.com</dnsName>
         <keyName>example-key-name</keyName>
         cproductCodesSet>
           <item>productCode>774F4FF8ductCode></item>
         <InstanceType>m1.small</InstanceType>
         <launchTime>2007-08-07T11:54:42.000Z</launchTime>
        </item>
      </instancesSet>
    </item>
  </reservationSet>
</DescribeInstancesResponse>
```

## **Related Operations**

- RunInstances
- <u>TerminateInstances</u>

# **DescribeKeyPairs**

The DescribeKeyPairs operation returns information about key pairs available to you. If you specify key pairs, information about those key pairs is returned. Otherwise, information for all registered key pairs is returned.

## **Request Parameters**

The following table describes the request parameters for DescribeKeyPairs. Parameter names are case sensitive.

Name	Description	Required
KeyName.n	Key pair IDs to describe. Type: String	No

#### **Response Elements**

The following table describes the default response tags included in DescribeKeyPairs responses.

Name	Description
keySet	A list of key pair descriptions.
	Type: <u>ec2:DescribeKeypairsResponseItemType</u> []

#### **Sample Request**

```
https://ec2.amazonaws.com/
?Action=DescribeKeyPairs
&KeyName.1=example-key-name
&...auth parameters...
```

#### **Sample Response**

## **Related Operations**

- <u>CreateKeypair</u>
- <u>DeleteKeypair</u>
- RunInstances

# DescribeSecurityGroups

The DescribeSecurityGroups operation returns information about security groups that you own.

If you specify security group names, information about those security group is returned. Otherwise, information for all security group is returned. If you specify a group that does not exist, a fault is returned.

## **Request Parameters**

The following table describes the request parameters for DescribeSecurityGroups. Parameter names are case sensitive.

Name	Description	Required
GroupName.n	List of security groups to describe. Type: String	No

#### **Response Elements**

The following table describes the default response tags included in DescribeSecurityGroups responses.

Name	Description
securityGroupInfo	Information about security groups.  Type: ec2:SecurityGroupItemType[]

#### **Sample Request**

https://ec2.amazonaws.com/ ?Action=DescribeSecurityGroups &GroupName.1=WebServers &GroupName.2=RangedPortsBySource

&...auth parameters...

#### **Sample Response**

```
<DescribeSecurityGroupsResponse xmlns="http://ec2.amazonaws.com/doc/2007</pre>
  <securityGroupInfo>
    <item>
      <ownerId>UYY3TLBUXIEON5NQVUUX60MPWBZIQNFM</ownerId>
      <groupName>WebServers
      <groupDescription>Web</groupDescription>
      <ipPermissions>
        <item>
         <ipProtocol>tcp</ipProtocol>
          <fromPort>80</fromPort>
          <toPort>80</toPort>
          <qroups/>
         <ipRanges>
            <item>
              <cidrIp>0.0.0.0/0</cidrIp>
            </item>
          </ipRanges>
         </item>
      </ipPermissions>
    </item>
    <item>
      <ownerId>UYY3TLBUXIEON5NQVUUX60MPWBZIQNFM</ownerId>
     <groupName>RangedPortsBySource
     <groupDescription>A</groupDescription>
      <ipPermissions>
        <item>
          <ipProtocol>tcp</ipProtocol>
         <fromPort>6000</fromPort>
          <toPort>7000</toPort>
          <qroups/>
          <ipRanges/>
        </item>
     </ip>
    </item>
 </securityGroupInfo>
</DescribeSecurityGroupsResponse>
```

#### **Related Operations**

- <u>CreateSecurityGroup</u>
- <u>AuthorizeSecurityGroupIngress</u>
- RevokeSecurityGroupIngress
- DeleteSecurityGroup

# GetConsoleOutput

The GetConsoleOutput operation retrieves console output for the specified instance.

Instance console output is buffered and posted shortly after instance boot, reboot, and termination. Amazon EC2 preserves the most recent 64 KB output which will be available for at least one hour after the most recent post.

## **Request Parameters**

The following table describes the request parameters for GetConsoleOutput. Parameter names are case sensitive.

Name	Description	Required
InstanceId	An instance ID returned from a previous call to RunInstances. Type: String	Yes

## **Response Elements**

The following table describes the default response tags included in GetConsoleOutput responses.

Name	Description
instanceId	The instance ID. Type: xsd:string
timestamp	The time the output was last updated. Type: xsd:dateTime
output	The console output, Base64 encoded. Type: xsd:string

#### **Sample Request**

https://ec2.amazonaws.com/ ?Action=GetConsoleOutput &InstanceId.1=i-2ea64347 &...auth parameters...

#### **Sample Response**

# ${\bf Modify Image Attribute}$

The ModifyImageAttribute operation modifies an attribute of an AMI.

#### **Attributes**

Attribute Name	Type	Description
launchPermission	List	Controls who has permission to launch the AMI. Launch permissions can be granted to specific users by adding userIds. To make the AMI public, add the all group.
productCodes	List	Associates a product code with AMIs. This allows developers to charge users for using AMIs. The user must be signed up for the product before they can launch the AMI. This is a write once attribute; after it is set, it cannot be changed or removed.

## **Request Parameters**

The following table describes the request parameters for ModifyImageAttribute. Parameter names are case sensitive.

Name	Description	Required
ImageId	AMI ID to modify. Type: String	Yes
Attribute	Specifies the attribute to modify. See the preceding attributes table for supported attributes.  Type: String	Yes
OperationType	Specifies the operation to perform on the attribute. See the preceding attributes table for supported operations for attributes.  Type: String Valid Values: add   remove Condition: Required for launchPermission	Conditional
UserId.n	User IDs to add to or remove from the launchPermission attribute.  Type: String  Condition: Required for launchPermission	Conditional
UserGroup. <mark>n</mark>	User groups to add to or remove from the launchPermission attribute. Currently, the all group is available, which will make it a public AMI. Type: String Condition: Required for launchPermission	Conditional
ProductCode.n	Attaches a product code to the AMI. Currently only one product code can be associated with an AMI. Once set, the product code cannot be changed or reset.  Type: String  Condition: Required for productCodes	Conditional

#### **Response Elements**

The following table describes the default response tags included in ModifyImageAttribute responses.

Name	Description
	true if the operation succeeded, otherwise false. Type: xsd:boolean

#### **Sample Request - Launch Permission**

https://ec2.amazonaws.com/ ?Action=ModifyImageAttribute &ImageId=ami-61a54008 &Attribute=launchPermission &OperationType=add &Group.1=all &UserId.1=495219933132 &...auth parameters...

#### **Sample Request - Product Codes**

https://ec2.amazonaws.com/ ?Action=ModifyImageAttribute &ImageId=ami-61a54008 &Attribute=productCodes &ProductCode.1=774F4FF8 &...auth parameters...

#### **Sample Response**

# **Related Operations**

- ResetImageAttribute
- <u>DescribeImageAttribute</u>

#### RebootInstances

The RebootInstances operation requests a reboot of one or more instances. This operation is asynchronous; it only queues a request to reboot the specified instance(s). The operation will succeed if the instances are valid and belong to the user. Requests to reboot terminated instances are ignored.

#### **Request Parameters**

The following table describes the request parameters for RebootInstance. Parameter names are case sensitive.

Name	Description	Required
InstanceId. <mark>n</mark>	One or more instance IDs. Type: String	Yes

#### **Response Elements**

The following table describes the default response tags included in RebootInstances responses.

Name	Description
result	true if the operation succeeded. Type: xsd:boolean

#### **Sample Request**

https://ec2.amazonaws.com/ ?Action=RebootInstances &InstanceId.1=i-2ea64347 &InstanceId.2=i-21a64348 &...auth parameters...

#### **Sample Response**

# RegisterImage

The RegisterImage operation registers an AMI with Amazon EC2. Images must be registered before they can be <u>launched</u>.

Each AMI is associated with an unique ID which is provided by the Amazon EC2 service through the RegisterImage operation. During registration, Amazon EC2 retrieves the specified image manifest from Amazon S3 and verifies that the image is owned by the user registering the image.

The image manifest is retrieved once and stored within the Amazon EC2. Any modifications to an image in Amazon S3 invalidates this registration. If you make changes to an image, <u>deregister</u> the previous image and register the new image.

## **Request Parameters**

The following table describes the request parameters for RegisterImage. Parameter names are case sensitive.

Name	Description	Required
ImageLocation	Full path to your AMI manifest in Amazon S3 storage. Type: String	Yes

## **Response Elements**

The following table describes the default response tags included in RegisterImage responses.

Name	Description
imageId	Unique ID of the newly registered machine image. Type: xsd:string

#### **Sample Request**

```
https://ec2.amazonaws.com/
?Action=RegisterImage
&ImageLocation=mybucket-myimage.manifest.xml
&...auth parameters...
```

#### **Sample Response**

# **Related Operations**

- <u>DescribeImages</u>
- <u>DeregisterImage</u>

# ResetImageAttribute

The  ${\tt ResetImageAttribute}$  operation resets an attribute of an AMI to its default value.



The productCodes attribute cannot be reset.

### **Request Parameters**

The following table describes the request parameters for ResetImageAttribute. Parameter names are case sensitive.

Name	Description	Required
ImageId	ID of the AMI for which an attribute will be described. Type: String	Yes
Attribute	Specifies the attribute to reset. Currently, only launchPermission is supported. In the case of launchPermission, all public and explicit launch permissions for the AMI are revoked.  Type: String	Yes

#### **Response Elements**

The following table describes the default response tags included in ResetImageAttribute responses.

Name	Description
return	true if the operation succeeded, otherwise false. Type: xsd:boolean

#### **Sample Request**

https://ec2.amazonaws.com/ ?Action=ResetImageAttribute &ImageId=ami-61a54008 &Attribute=launchPermission &...auth parameters...

#### **Sample Response**

<ResetImageAttributeResponse xmlns="http://ec2.amazonaws.com/doc/2007-08
<return>true</return>

</ResetImageAttributeResponse>

## **Related Operations**

- ModifyImageAttribute
- <u>DescribeImageAttribute</u>

## RevokeSecurityGroupIngress

The RevokeSecurityGroupIngress operation revokes permissions from a security group. The permissions used to revoke must be specified using the same values used to grant the permissions.

Permissions are specified by IP protocol (TCP, UDP, or ICMP), the source of the request (by IP range or an Amazon EC2 user-group pair), the source and destination port ranges (for TCP and UDP), and the ICMP codes and types (for ICMP).

Permission changes are quickly propagated to instances within the security group. However, depending on the number of instances in the group, a small delay is might occur, .

When revoking a user/group pair permission, <code>GroupName</code>, <code>SourceSecurityGroupName</code> and <code>SourceSecurityGroupOwnerId</code> must be specified. When authorizing a CIDR IP permission, <code>GroupName</code>, <code>IpProtocol</code>, <code>FromPort</code>, <code>ToPort</code> and <code>CidrIp</code> must be specified. Mixing these two types of parameters is not allowed.

### **Request Parameters**

The following table describes the request parameters for RevokeSecurityGroupIngress. Parameter names are case sensitive.

Name	Description	Required
GroupName	Name of the group to modify. Type: String	Yes
SourceSecurityGroupName	Name of security group to revoke access to when operating on a user/group pair.  Type: String  Condition: Required when revoking user/group pair permission.	Conditional
SourceSecurityGroupOwnerId	Owner of security group to revoke access to when operating on a user/group pair.  Type: String Condition: Required when revoking user/group pair permission.	Conditional
IpProtocol	IP protocol to revoke access to when operating on a CIDR IP.  Type: String Valid Values: tcp   udp   icmp Condition: Required when revoking CIDR IP permission.	Conditional
FromPort	Bottom of port range to revoke access to when operating on a CIDR IP. This contains the ICMP type if ICMP is being authorized.  Type: Int Condition: Required when revoking CIDR IP permission.	Conditional
ToPort	Top of port range to revoke access to when operating on a CIDR IP. This contains the ICMP code if ICMP is being authorized.  Type: Int  Condition: Required when revoking CIDR IP permission.	Conditional
CidrIp	CIDR IP range to revoke access to when operating on a CIDR IP.  Type: String Condition: Required when revoking CIDR IP permission.	Conditional

#### **Response Elements**

The following table describes the default response tags included in RevokeSecurityGroupIngress responses.

Name	Description
return	true if permissions successfully revoked. Type: xsd:boolean

#### **Sample Request**

```
https://ec2.amazonaws.com/
?Action=AuthorizeSecurityGroupIngress
&IpProtocol=tcp
&FromPort=80
&ToPort=80
&CidrIp=0.0.0.0/0
&...auth parameters...
```

#### **Sample Response**

<RevokeSecurityGroupIngressResponse xmlns="http://ec2.amazonaws.com/doc/
 <return>true</return>
 </RevokeSecurityGroupIngressResponse>

### **Related Operations**

- <u>CreateSecurityGroup</u>
- <u>DescribeSecurityGroups</u>
- <u>AuthorizeSecurityGroupIngress</u>
- DeleteSecurityGroup

#### **RunInstances**

The RunInstances operation launches a specified number of instances.



The Query version of RunInstances only allows instances of a single AMI to be launched in one call. This is different from the SOAP API version of the call, but similar to the ec2-run-instances command line tool.

If Amazon EC2 cannot launch the minimum number AMIs you request, no instances launch. If there is insufficient capacity to launch the maximum number of AMIs you request, Amazon EC2 launches as many as possible to satisfy the requested maximum values.

Every instance is launched in a <u>security group</u>. If you do not specify a security group at launch, the instances start in your default security group.

An optional instance type can be specified. For information about instance types, see <u>Selecting Instance Types</u>.

You can provide an optional <u>key pair ID</u> for each image in the launch request. All instances that are created from images that use this key pair will have access to the associated public key at boot. You can use this key to provide secure access to an instance of an image on a per-instance basis. Amazon EC2 public images use this feature to provide secure access without passwords.



#### ) Important

Launching public images without a key pair ID will leave them inaccessible.

The public key material is made available to the instance at boot time by placing it in the openssh\_id.pub file on a logical device that is exposed to the instance as /dev/sda2 (the ephemeral store). The format of this file is suitable for use as an entry within ~/.ssh/authorized\_keys (the OpenSSH format). This can be done at boot (e.g., as part of rc.local) allowing for secure access without passwords.

Optional user data can be provided in the launch request. All instances

comprising the launch request have access to this data (see **Instance Metadata** for details).



If any of the AMIs have a product code attached for which the user has not subscribed, the RunInstances call will fail.

### **Request Parameters**

The following table describes the request parameters for RunInstances. Parameter names are case sensitive.

Name	Description	Required
ImageId	ID of the AMI with which to launch instances. Type: String	Yes
MinCount	Minimum number of instances to launch.  Type: Int	Yes
MaxCount	Maximum number of instances to launch. Type: Int	Yes
KeyName	Name of the key pair with which to launch instances. Type: String	No
SecurityGroup. <mark>n</mark>	Names of the security groups with which to associate the instances.  Type: String	No
UserData	The user data available to the launched instances. This should be Base64 encoded. For more information on encoding details, see <a href="UserDataType">UserDataType</a> . Type: String	No
AddressingType	The addressing scheme with which to launch the instance. The supported addressing type is public. For the public scheme, the instance has a private and public IP address that are mapped through NAT. For more information, see <a href="Instance Addressing">Instance Addressing</a> .	No
	To support our legacy cluster,	
	early beta customers can also	
	specify direct. In the direct	
	scheme, the instance has a	
	single public IP address.	
	Type: String	
InstanceType	This specifies the instance type.  Options include m1.small, m1.large, and m1.xlarge.  The default value is m1.small.  For more information on instance types, see Selecting Instance Types  Type: xsd:string	No

#### **Response Elements**

The following table describes the default response tags included in RunInstances responses.

Name	Description
RunInstancesResponse	Status information about the instances launched. Type: <a href="mailto:ec2ReservationInfoType">ec2ReservationInfoType</a>

#### **Sample Request**

https://ec2.amazonaws.com/ ?Action=RunInstances &ImageId=ami-60a54009 &MaxCount=3 &MinCount=1 &AddressingType=public &...auth parameters...

#### **Sample Response**

```
<RunInstancesResponse xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
 <reservationId>r-47a5402e</reservationId>
 <ownerId>495219933132/ownerId>
 <qroupSet>
    <item>
      <groupId>default
    </item>
 </groupSet>
 <instancesSet>
    <item>
      <instanceId>i-2ba64342</instanceId>
      <imageId>ami-60a54009</imageId>
      <instanceState>
        <code>0</code>
        <name>pending</name>
      </instanceState>
      <privateDnsName></privateDnsName>
      <dnsName></dnsName>
      <keyName>example-key-name</keyName>
       <amiLaunchIndex>0</amiLaunchIndex>
      <InstanceType>m1.small</InstanceType>
      <launchTime>2007-08-07T11:51:50.000Z</launchTime>
    </item>
    <item>
      <instanceId>i-2bc64242</instanceId>
      <imageId>ami-60a54009/imageId>
      <instanceState>
        <code>0</code>
        <name>pending</name>
      </instanceState>
      <privateDnsName></privateDnsName>
      <dnsName></dnsName>
     <keyName>example-key-name</keyName>
      <amiLaunchIndex>1</amiLaunchIndex>
      <InstanceType>m1.small</InstanceType>
      <launchTime>2007-08-07T11:51:50.000Z</launchTime>
    </item>
    <item>
      <instanceId>i-2be64332</instanceId>
      <imageId>ami-60a54009</imageId>
      <instanceState>
        <code>0</code>
        <name>pending</name>
      </instanceState>
```

#### **Related Operations**

- <u>DescribeInstances</u>
- <u>TerminateInstances</u>
- <u>AuthorizeSecurityGroupIngress</u>
- RevokeSecurityGroupIngress
- <u>DescribeSecurityGroups</u>

#### **TerminateInstances**

The TerminateInstances operation shuts down one or more instances. This operation is idempotent; if you terminate an instance more than once, each call will succeed.

Terminated instances will remain visible after termination (approximately one hour).

#### **Request Parameters**

The following table describes the request parameters for TerminateInstances. Parameter names are case-sensitive.

Name	Description	Required
InstanceId.n	One or more instance IDs returned. Type: String	Yes

### **Response Elements**

The following table describes the default response tags included in TerminateInstances responses.

Name	Description
instancesSet	A complex type describing the current and new state of each instance specified. Type: <a href="ec2:TerminateInstancesResponseInfoType">ec2:TerminateInstancesResponseInfoType</a> []

#### **Sample Request**

https://ec2.amazonaws.com/ ?Action=TerminateInstances &InstanceId.1=i-2ea64347 &InstanceId.2=i-21a64348 &...auth parameters...

#### **Sample Response**

```
<TerminateInstancesResponse xmlns="http://ec2.amazonaws.com/doc/2007-08-</pre>
 <instancesSet>
    <item>
      <instanceId>i-28a64341</instanceId>
      <shutdownState>
        <code>32</code>
        <name>shutting-down</name>
      </shutdownState>
      ousState>
        <code>16</code>
        <name>running</name>
      </previousState>
    </item>
    <item>
      <instanceId>i-21a64348</instanceId>
      <shutdownState>
        <code>32</code>
        <name>shutting-down</name>
      </shutdownState>
      ousState>
        <code>16</code>
        <name>running</name>
      </previousState>
    </item>
 </instancesSet>
</TerminateInstancesResponse>
```

## **Related Operations**

• <u>DescribeInstances</u>

## **Command Line Tools Reference**

#### Introduction

The Amazon EC2 command line tools provide a command line interface to the web service API. This section describes each tool and its command line arguments in detail.

Command line options and arguments are based on the GNU getopt conventions. Parameters are invoked using flags, which typically come in short and long form. In their short form, flags use a single character preceded by a dash. In their long form, flags use a more expressive name preceded by two dashes.



Some common options apply to all command line tools. These are described below and are not included in the description of the specific tools.

#### **Errors**

Any service errors encountered by the command line tools are passed straight through from the API. For more information about these errors, see <u>API Error Codes</u>.

#### **Common Options**

Most command line tools described in this section accept the the following set of optional parameters:

Option	Description
-U URL	URL is the uniform resource locator of the Amazon EC2 web service entry point. Default: The EC2_URL environment variable, or http://ec2.amazonaws.com if the environment variable is not set.  Example: -U http://ec2.amazonaws.com
-K EC2- PRIVATE- KEY	The private key to use when constructing requests to Amazon EC2.  Default: The value of the EC2_PRIVATE_KEY environment variable.  Example: -K pk-HKZYKTAIG2ECMXYIBH3HXV4ZBZQ55CL0.pem
-C EC2- CERT	The X.509 certificate to use when constructing requests to Amazon EC2.  Default: The value of the EC2_CERT environment variable.  Example: -C cert-HKZYKTAIG2ECMXYIBH3HXV4ZBZQ55CL0.pem
- <i>v</i>	Displays verbose output by showing the SOAP request and response on the command line. This is particularly useful if you are building tools to talk directly to our SOAP API.
show- empty- fields	Shows empty columns as (nil).
debug	Prints internal debugging information. This is useful to assist us when troubleshooting problems.
-?	Displays help.
-	If - is specified as an argument to one of the parameters, a list of arguments are read from standard input. This is useful for piping the output of one command into the input of another.
	Example: ec2-describe-instances   grep running   cut -f 2   ec2-terminate-instances -i -

## **AMI Tools**

This section describes each tool used to create AMIs and its command line arguments in detail.

# ec2-bundle-image

#### **Syntax**

**ec2-bundle-image** -k  $private\_key$  -c  $ec2\_cert$  -u  $user\_id$  -i  $image\_path$  -r  $\{i386 \mid x86\_64\}$  [-d destination] [-p  $ami\_prefix$ ] [-b, --batch]

#### **Description**

Create a bundled AMI from an operating system image created in a loopback file. For more information, see Creating an AMI through a Loopback File.



Scripts that require a copy of the public key from the launch key pair must obtain the key from the instance's metadata (not the key file in the ephemeral store) for instances bundled with the 2007-08-29 AMI tools and later. AMIs bundled before this release will continue to work normally.

### **Options**



This tool does not support the **common options** 

Option	Description	Required
-k,privatekey private_key	The path to the user's PEM-encoded RSA key file. Example: -k pk-HKZYKTAIG2ECMXYIBH3HXV4ZBZQ55CL0.pem	Yes
-c,cert ec2_cert	The user's PEM encoded RSA public key certificate file. Example: -c cert-HKZYKTAIG2ECMXYIBH3HXV4ZBZQ55CLO.pem	Yes
-u,user user_id	The user's AWS account number without dashes. Do not use the Access Key ID. Example: -u 495219933132	Yes
-i,image image_path	The path to the image to bundle.  Example: -i /var/spool/my-image/version-2/debian.img	Yes
-r,arch {i386   x86_64}	Specifies 32-bit (m1.small) or 64-bit architecture (m1-large and m1-xlarge).  Default: None Example: -r x86_64	Yes
-d,destination destination	The directory in which to create the bundle.  Default: The current directory  Example: -d /var/run/my-bundle	No
-p,prefix ami_prefix	The filename prefix for bundled AMI files.  Default: image Example: -p my-image-is-special	No
help	Display the help message. Example:help	No
manual	Display the manual entry. Example:manual	No
-b,batch	Runs without interaction and suppresses all warnings. Will attempt to automatically determine architecture. Example: -b	No

### Output

Status messages describing the stages and status of the bundling process.

#### **Example**

This example creates a bundled AMI from an operating system image that was created in a loopback file:

```
$ ec2-bundle-image -k pk-HKZYKTAIG2ECMXYIBH3HXV4ZBZQ55CL0.pem -c cert-HK
Splitting bundled/fred.gz.crypt...
Created fred.part.00
Created fred.part.01
Created fred.part.02
Created fred.part.03
Created fred.part.04
Created fred.part.05
Created fred.part.06
Created fred.part.07
Created fred.part.08
Created fred.part.09
Created fred.part.10
Created fred.part.11
Created fred.part.12
Created fred.part.13
Created fred.part.14
Generating digests for each part...
Digests generated.
Creating bundle manifest...
Bundle Image complete.
```

### **Related Topics**

- ec2-bundle-vol
- ec2-unbundle
- ec2-upload-bundle
- ec2-download-bundle
- ec2-delete-bundle

### ec2-bundle-vol

#### **Syntax**

**ec2-bundle-vol** -k private\_key -u user\_id -c ec2\_cert -r  $\{i386 \mid x86\_64\}$  [-s size] [-d destination] [-e exclude\_directory\_1,exclude\_directory\_1,...] [-p ami\_prefix] [-v volume] [--ec2cert ami\_path] [--fstab fstab\_path] [-b, --batch]

#### **Description**

Creates a bundled AMI by compressing, encrypting and signing a snapshot of the local machine's root file system.



Scripts that require a copy of the public key from the launch key pair must obtain the key from the instance's metadata (not the key file in the ephemeral store) for instances bundled with the 2007-08-29 AMI tools and later. AMIs bundled before this release will continue to work normally.

### **Options**

#### Note

This tool does not support the **common arguments** 

Option	Description	Required
-k,privatekey private_key	The path to the user's PEM-encoded RSA key file. Example: -k pk- HKZYKTAIG2ECMXYIBH3HXV4ZBZQ55CL0.pem	Yes
-c,cert <i>ec2_cert</i>	The user's PEM encoded RSA public key certificate file.  Example: -c cert- HKZYKTAIG2ECMXYIBH3HXV4ZBZQ55CL0.pem	Yes
-u,user <i>user_id</i>	The user's AWS account number without dashes. Do not use the Access Key ID. Example: -u 495219933132	Yes
-r,arch {i386   x86_64}	Specifies 32-bit (m1.small) or 64-bit architecture (m1-large and m1-xlarge). Example: -r i386	Yes
-s,size <i>size</i>	The size, in MB (1024 * 1024 bytes), of the image file to create. The maximum size is 10240 MB.  Default: 10240  Example: -s 2048	No
-d,destination destination	The directory in which to create the bundle.  Default: /tmp  Example: -d /var/run/my-bundle	No
-e,exclude directory_1,directory_2,	A list of absolute directory paths to exclude from the bundle operation. This overrides theall parameter. Example: -e /tmp,/home/secret-data	No
-p,prefix ami_prefix	The filename prefix for bundled AMI files.  Default: image  Example: -p my-image-is-special	No
-v,volume <i>volume</i>	The absolute path to the mounted volume from which to create the bundle.  Default: The root directory (/)  Example: -v /mnt/my-customized-ami	No
-a,all	Bundle all directories, including those on remotely mounted filesystems.  Example: -a	No
ec2cert ami_path	The path to the Amazon EC2 X509 public key	No

	certificate.  Default: /etc/aes/amiutil/cert-ec2.pem  Example:ec2cert /etc/aes/amiutil/cert-ec2.pem	
help	Display the help message. Example:help	No
manual	Display the manual entry. Example:manual	No
fstab fstab_path	The path to the fstab to bundle into the image. If this is not specified, Amazon EC2 will provide its own fstab.  Example:fstab /etc/fstab	No
-b,batch	Runs without interaction and suppresses all warnings. Will attempt to automatically determine architecture. Example: -b	No

### Output

Status messages describing the stages and status of the bundling

#### **Example**

This example creates a bundled AMI by compressing, encrypting and signing a snapshot of the local machine's root file system.

```
$ ec2-bundle-vol -d /mnt -k pk-HKZYKTAIG2ECMXYIBH3HXV4ZBZQ55CL0.pem -c c
Copying / into the image file /mnt/image.img...
Excluding:
     sys
     dev/shm
     proc
     dev/pts
     proc/sys/fs/binfmt_misc
     dev
    media
    mnt
     proc
    sys
     tmp/image.img
    mnt/img-mnt
1+0 records in
1+0 records out
mke2fs 1.38 (30-Jun-2005)
warning: 256 blocks unused.
Splitting /mnt/image.gz.crypt...
Created image.part.00
Created image.part.01
Created image.part.02
Created image.part.03
Created image.part.22
Created image.part.23
Generating digests for each part...
Digests generated.
Creating bundle manifest...
Bundle Volume complete.
```

### **Related Topics**

- ec2-bundle-image
- ec2-unbundle
- ec2-upload-bundle
- ec2-download-bundle
- ec2-delete-bundle

## ec2-delete-bundle

### **Syntax**

ec2-delete-bundle -b s3\_bucket -a access\_key\_id -s secret\_key [-m
manifest\_path] [-p ami\_prefix] [--url url] [--retry] [-y]

### **Description**

Deletes the specified bundle from Amazon S3 storage.

### **Options**



This tool does not support the **common arguments** 

Option	Description	Required
-b,bucket s3_bucket	The name of the Amazon S3 bucket containing the bundled AMI Example: -b aes-cracker-ami-bucket	Yes
-a,access-key access_key_id	The AWS access key ID. Example: -a 10QMXFEV71ZS32XQFTR2	Yes
-s,secret-key secret_key	The AWS secret access key. Example: -s DMADSSfPfdaDjbK+RRUhS/aDrjsiZadgAUm8gRU2	Yes
-m,manifest manifest_path	The path to the unencrypted manifest file. Example: -m /var/spool/my-first-bundle/Manifest	No
-p,prefix ami_prefix	The bundled AMI filename prefix. Example: -p eos-	No
url <i>url</i>	The Amazon S3 service URL.  Default: https://s3.amazonaws.com  Example:url https://s3.amazonaws.ie	No
retry	Automatically retries failed uploads. Use with caution. Example:retry	No
-y,yes	Automatically assumes the answer to all prompts is 'yes'. Example: -y	No
help	Display the help message. Example:help	No
manual	Display the manual entry. Example:manual	No

### Output

Amazon EC2 displays status messages indicating the stages and status of the delete process.

#### **Example**

This example deletes a bundle from Amazon S3:

```
$ ec2-delete-bundle -b my-s3-bucket -a 10QMXFEV71ZS32XQFTR2 -s DMADSSfPf
Deleting files:
my-s3-bucket/fred.manifest.xml
my-s3-bucket/fred.part.00
my-s3-bucket/fred.part.01
my-s3-bucket/fred.part.02
my-s3-bucket/fred.part.03
my-s3-bucket/fred.part.04
my-s3-bucket/fred.part.05
my-s3-bucket/fred.part.06
Continue? [y/n]
Deleted my-s3-bucket/fred.manifest.xml
Deleted my-s3-bucket/fred.part.00
Deleted my-s3-bucket/fred.part.01
Deleted my-s3-bucket/fred.part.02
Deleted my-s3-bucket/fred.part.03
Deleted my-s3-bucket/fred.part.04
Deleted my-s3-bucket/fred.part.05
Deleted my-s3-bucket/fred.part.06
ec2-delete-bundle complete.
```

### **Related Topics**

- ec2-bundle-image
- ec2-bundle-vol
- ec2-unbundle
- ec2-upload-bundle
- ec2-download-bundle

## ec2-download-bundle

#### **Syntax**

ec2-download-bundle -b s3\_bucket -m manifest -a access\_key\_id -s
secret\_key -k private\_key [-p ami\_prefix] [-d directory] [--url url]

### **Description**

Download the specified bundles from S3 storage.

### **Options**



This tool does not support the **common arguments** 

Option	Description	Required
b,bucket -s3_bucket	The name of the Amazon S3 bucket where the bundle is located.  Example: -b aes-cracked	Yes
-m,manifest manifest	The manifest path and filename.  Example: -m /var/spool/my-first-bundle/Manifest	Yes
-a,access-key access_key_id	Your AWS access key ID. Example: -a 10QMXFEV71ZS32XQFTR2	Yes
-s,secret-key secret_key	Your AWS secret access key. Example: -s DMADSSfPfdaDjbK+RRUhS/aDrjsiZadgAUm8gRU2	Yes
-k,privatekey private_key	The private key used to decrypt the manifest.  Example: -k pk-HKZYKTAIG2ECMXYIBH3HXV4ZBZQ55CL0.pem	Yes
-p,prefix ami_prefix	The filename prefix for the bundled AMI files.  Default: image  Example: -p my-image	No
-d,directory directory	The directory where the downloaded bundles are saved. The directory must exist.  Default: The current working directory  Example: -d /tmp/my-downloaded-bundle	No
url <i>url</i>	The S3 service URL. Default: https://s3.amazonaws.com Example:url https://s3.amazonaws.ie	No
help	Display the help message. Example:help	No

### Output

Status messages indicating the various stages of the download process are displayed.

#### **Example**

This example downloads creates the bundled directory and downloads the bundle from the my-s3-bucket Amazon S3 bucket.

```
$ mkdir bundled
$ ec2-download-bundle -b my-s3-bucket -m fred.manifest.xml -a 10QMXFEV71
downloading manifest https://s3.amazonaws.com/my-s3-bucket/image.manifest.xml to bundled/i
downloading part https://s3.amazonaws.com/my-s3-bucket/image.part.00 to bundled/image.part
Downloaded image.part.00 from https://s3.amazonaws.com/my-s3-bucket.
downloading part https://s3.amazonaws.com/my-s3-bucket/image.part.01 to bundled/image.part
Downloaded image.part.01 from https://s3.amazonaws.com/my-s3-bucket.
downloading part https://s3.amazonaws.com/my-s3-bucket/image.part.02 to bundled/image.part
Downloaded image.part.02 from https://s3.amazonaws.com/my-s3-bucket.
downloading part https://s3.amazonaws.com/my-s3-bucket/image.part.03 to bundled/image.part
Downloaded image.part.03 from https://s3.amazonaws.com/my-s3-bucket.
downloading part https://s3.amazonaws.com/my-s3-bucket/image.part.04 to bundled/image.part
Downloaded image.part.04 from https://s3.amazonaws.com/my-s3-bucket.
downloading part https://s3.amazonaws.com/my-s3-bucket/image.part.05 to bundled/image.part
Downloaded image.part.05 from https://s3.amazonaws.com/my-s3-bucket.
downloading part https://s3.amazonaws.com/my-s3-bucket/image.part.06 to bundled/image.part
Downloaded image.part.06 from https://s3.amazonaws.com/my-s3-bucket.
Download Bundle complete.
```

### **Related Topics**

- ec2-bundle-image
- <u>ec2-bundle-vol</u>
- ec2-unbundle
- ec2-upload-bundle
- ec2-delete-bundle

### ec2-unbundle

### **Syntax**

ec2-unbundle -m manifest -k private\_key [-d destination\_directory] [-s
source\_directory]

### **Description**

Recreates the AMI from the bundled AMI parts.

### **Options**



This tool does not support the **common arguments** 

Option	Description	Required
-m,manifest manifest	The path to the unencrypted AMI manifest file. Example: -m /var/spool/my-first-bundle/Manifest	Yes
-k,privatekey private_key	The path to your PEM-encoded RSA key file. Example: -k \$HOME/pk-234242DEADCAFE.pem	Yes
-d,destination destination_directory	The directory in which to unbundle the AMI. The destination directory must exist.  Default: The current directory  Example: -d /tmp/my-image	No
-s,source source_directory	The directory containing the bundled AMI parts.  Default: The current directory  Example: -s /tmp/my-bundled-image	No
help	Display the help message. Example:help	No

#### **Example**

This example unbundles the AMI specified in the fred.manifest.xml file:

```
$ mkdir unbundled
$ ec2-unbundle -m fred.manifest.xml -s bundled -d unbundled
cat bundled/fred.part.00 bundled/fred.part.01 bundled/fred.part.02 bundled/fred.part.03 t
Unbundle complete.
$ ls -l unbundled
total 1025008
-rw-r--r-- 1 root root 1048578048 Aug 25 23:46 fred.img
```

### Output

Status messages indicating the various stages of the unbundling process are displayed.

### **Related Topics**

- ec2-bundle-image
- ec2-bundle-vol
- ec2-upload-bundle
- ec2-download-bundle
- ec2-delete-bundle

# ec2-upload-bundle

#### **Syntax**

ec2-upload-bundle -b s3\_bucket -m manifest -a access\_key\_id -s
secret\_key [--acl ac1] [--ec2cert certificate] [-d directory] [--part part] [-url ur1] [--retry] [--skipmanifest]

### **Description**

Upload a bundled AMI to Amazon S3 storage.

### **Options**



This tool does not support the **common options** 

Option	Description	Required
-b,bucket s3_bucket	The name of the Amazon S3 bucket in which to store the bundle. If the bucket doesn't exist it will be created (if the bucket name is available). Example: -b aes-cracker-ami	Yes
-m,manifest manifest	The path to the manifest file. The manifest file is created during the bundling process and can be found in the directory containing the bundle.  Example: -m /var/spool/my-first-bundle/Manifest	Yes
-a,access- key access_key_id	Your AWS access key ID. Example: -a 10QMXFEV71ZS32XQFTR2	Yes
-s,secret- key <i>secret_key</i>	Your AWS secret access key. Example: -s DMADSSfPfdaDjbK+RRUhS/aDrjsiZadgAUm8gRU2	Yes
acl <i>acl</i>	The access control list policy of the bundled image.  Valid Values: public-read   aws-exec-read  Default: aws-exec-read  Example:acl public-read	No
ec2cert certificate	The path to the Amazon EC2 X509 public key certificate.  Default: /etc/aes/amiutil/cert-ec2.pem  Example:ec2cert /etc/aes/amiutil/cert-ec2.pem	No
-d,directory directory	The directory containing the bundled AMI parts.  Default: The directory containing the manifest file (see the -m option).  Example: -d /var/run/my-bundle	No
part <i>part</i>	Starts uploading the specified part and all subsequent parts.  Example:part	No
url <i>url</i>	The S3 service URL.  Default: https://s3.amazonaws.com  Example:url https://s3.amazonaws.ie	No
retry	Automatically retries failed uploads. Use with caution.  Example:retry	No
skipmanifest	Does not upload the manifest. Example:skipmanifest	No
help	Display the help message. Example:help	No

manual	Display the manual entry.	No
	Example:manual	

### Output

Amazon EC2 displays status messages that indicate the stages and status of the upload process.

#### **Example**

This example uploads the bundle specified by the bundled/fred.manifest.xml manifest:

```
$ ec2-upload-bundle -b my-s3-bucket -m bundled/fred.manifest.xml -a 10QM
Encrypting bundle manifest...
Completed encryption.
Uploading encrypted manifest...
Uploaded encrypted manifest to http://s3.amazonaws.com:80/alpowell-images/fred.manifest.xn
Uploading bundled AMI parts to http://s3.amazonaws.com:80/alpowell-images...
Uploaded fred.part.00 to http://s3.amazonaws.com:80/alpowell-images/fred.part.00.
Uploaded fred.part.01 to http://s3.amazonaws.com:80/alpowell-images/fred.part.01.
Uploaded fred.part.02 to http://s3.amazonaws.com:80/alpowell-images/fred.part.02.
Uploaded fred.part.03 to http://s3.amazonaws.com:80/alpowell-images/fred.part.03.
Uploaded fred.part.04 to http://s3.amazonaws.com:80/alpowell-images/fred.part.04.
Uploaded fred.part.05 to http://s3.amazonaws.com:80/alpowell-images/fred.part.05.
Uploaded fred.part.06 to http://s3.amazonaws.com:80/alpowell-images/fred.part.06.
Uploaded fred.part.07 to http://s3.amazonaws.com:80/alpowell-images/fred.part.07.
Uploaded fred.part.08 to http://s3.amazonaws.com:80/alpowell-images/fred.part.08.
Uploaded fred.part.09 to http://s3.amazonaws.com:80/alpowell-images/fred.part.09.
Uploaded fred.part.10 to http://s3.amazonaws.com:80/alpowell-images/fred.part.10.
Uploaded fred.part.11 to http://s3.amazonaws.com:80/alpowell-images/fred.part.11.
Uploaded fred.part.12 to http://s3.amazonaws.com:80/alpowell-images/fred.part.12.
Uploaded fred.part.13 to http://s3.amazonaws.com:80/alpowell-images/fred.part.13.
Uploaded fred.part.14 to http://s3.amazonaws.com:80/alpowell-images/fred.part.14.
Upload Bundle complete.
```

### **Related Topics**

- ec2-bundle-image
- ec2-bundle-vol
- ec2-unbundle
- ec2-download-bundle
- ec2-delete-bundle

# **API Tools**

This section describes each API tool and its command line arguments in detail.

## **List of Operations by Function**

#### **Images**

- ec2-register
- ec2-deregister
- ec2-describe-images

#### **Instances**

- ec2-run-instances
- ec2-describe-instances
- ec2-terminate-instances
- ec2-confirm-product-instance

#### **Key Pairs**

- <u>ec2-add-keypair</u>
- ec2-describe-keypairs
- ec2-delete-keypair
- ec2-fingerprint-key

#### **Image Attributes**

- ec2-modify-image-attribute
- ec2-describe-image-attribute
- <u>ec2-reset-image-attribute</u>

#### **Security Groups**

- ec2-add-group
- ec2-delete-group
- ec2-describe-groups
- <u>ec2-authorize</u>
- <u>ec2-revoke</u>

# ec2-add-group

### **Syntax**

ec2-add-group group -d description

## Description

Creates a new security group. Group names must be unique per account.

### **Options**

Option	Description	Required
group	Name of the security group. Example: webservers	Yes
-d description	Description of the group. This is informational only. If the description contains spaces, you must enclose it in single quotes ('). Example: -d 'Web servers'	Yes

### Output

Amazon EC2 returns a table that contains the following information:

- Output type identifier ("GROUP").
- Group name.
- Group description.

Amazon EC2 displays errors on stderr.

### **Example**

This example creates the websrv security group.

```
$ ec2-add-group websrv -d 'Web servers'
GROUP websrv Web servers
```

### **Related Topics**

- <u>CreateSecurityGroup</u>
- ec2-describe-groups
- ec2-delete-group
- ec2-authorize
- <u>ec2-revoke</u>

# ec2-add-keypair

### **Syntax**

ec2-add-keypair *key* 

### **Description**

Creates a new 2048 bit RSA key pair with the specified name. The public key is stored by Amazon EC2 and the private key is displayed on the console. The private key is returned as an unencrypted PEM encoded PKCS#8 private key. If a key with the specified name already exists, Amazon EC2 returns an error.

## **Options**

Option	Description	Required
key	Name of the key pair. Example: mysecretkey	Yes

#### Output

Amazon EC2 returns a table that contains the following information:

- Output type identifier ("KEYPAIR").
- Key pair name.
- Private key fingerprint.
- Private key. This value is displayed on a new line.

Amazon EC2 displays errors on stderr.

#### **Example**

This example creates a key pair named gsg-keypair:

```
$ ec2-add-keypair gsg-keypair
KEYPAIR gsg-keypair 1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:f1:6f
  ---BEGIN RSA PRIVATE KEY----
MIIEoOIBAAKCAOBuLFq5ujHrtm1jnutSuoO8Xe56LlT+HM8v/xkaa39EstM3/aFxTHqElOiJLChp
HungXQ29VTc8rc1bW0lkdi230H5eqkMHGhvEwga0HWASUMll4o3o/IX+0f2UcPoKC0VUR+jx71Sq
5AU52EQfanIn3ZQ8lFW7Edp5a3q4DhjGlUKToHVbicL5E+g45zfB95wIyywWZfeW/UUF3LpGZyq/
ebIUlq1qTbHkLbCC2r7RTn8vpQwp47BGVYGtGSBMpTRP5hnbzzuqj3itkiLHjU39S2sJCJ0TrJx5
i8BygR4s3mHKBj8l+ePQxG1kGbF6R4yg6sECmXn17MRQVXODNHZbAgMBAAECggEAY1tsiUsIwDl5
91CXirkYGuVfLyLflXenxfI50mDFms/mumTqloHO7tr0oriHDR5K7wMcY/YY5YkcXNo7mvUVD1pM
ZNUJs7rw9gZRTrf7LylaJ58k0cyajw8TsC4e4LPbFaHwS1d6K8rXh64o6WgW4SrsB6ICmr1kGQ17
3wcfgt5ecIu4TZf00E9IHjn+2eRlsrjBde0Ri7KiUNC/pAG23I6MdD0FEQRcCSigCj+4/mciFUSA
SWS4dMbrpb9FNSIcf9dcLxVM7/6KxgJNfZc9XWzUw77Jg8x92Zd0fVhH0ux5IZC+UvSKWB4dyfcI
tE8C3p9bbU9VGyY5vLCAiIb4qQKBgQDLi024GXrIkswF32YtBBMuVgLGCwU9h9Hl09mKAc2m8Cm1
jUE51pzRjTedc912qiIMUTwtqnw42auSCzbUeYMURPtDqyQ7p6AjMujp9EPemcSVOK9vXYL0Ptco
xW9MC0dtV6iPkCN7q0qiZXPRKaFbWADp16p8UAIvS/a5XXk5jwKBqQCKkpHi2EISh1uRkhxljyWC
iDCiK6JBRsMvpLbc0v5dKwP5alo1fmdR5PJaV2qvZSj5CYNpMAy1/EDNTY50SIJU+0KFmQbyhsbm
rdLNLDL4+TcnT7c62/aH01ohYaf/VCbRhtLlBfqGoQc7+sAc8vmKkesnF7CqCEKDyF/dhrxYdQKB
gC0iZzzNAapayz1+JcVTwwEid6j9JqNXbBc+Z2YwMi+T0Fv/P/hwkX/ype0XnIUcw0Ih/YtGBVAC
DQbsz7LcY1HqXiHKYNWNvXgwwO+oiChjxvEkSdsTTIfnK4VSCvU9BxDbQHjdiNDJbL6oar92UN7V
rBYvChJZF7LvUH4YmVpHAoGAbZ2X7XvoeE0+uZ58/BGK0IGHByHBDiXtzMhdJr15HTYjxK70gTZm
gK+8zp4L9IbvLGDMJ08vft32XPEWuvI8twCzFH+CsWLQADZMZKSsBas0Z/h1FwhdMgCMcY+Qlzd4
JZKjTSu3i7vhvx6RzdSedXEMNTZWN4qlIx3kR5aHcukCgYA9T+Zrvm1F0seQPbLknn7EqhXIjBaT
P8TTvW/6bdPi23ExzxZn7KOdrfclYRph1LHMpAONv/x2xALIf91UB+v5ohy1oDoasL0qij1houRe
2ERKKdwz0ZL9SWq6VTdhr/5G994CK72fy5WhyERbDjUIdHaK3M849JJuf8cSrvSb4g==
----END RSA PRIVATE KEY----
```

### **Related Topics**

- <u>CreateKeypair</u>
- ec2-describe-keypairs
- <u>ec2-delete-keypair</u>

## ec2-authorize

### **Syntax**

ec2-authorize group [-P protocol] (-p port\_range | -t icmp\_type\_code) [-u
source\_group\_user ...] [-o source\_group ...] [-s source\_subnet ...]

### **Description**

Adds a rule to the specified security group. If no source host, group or subnet is provided, requests from any source address will be honored.

### **Options**

Option	Description	Required
group	The group to which this rule will apply. Example: webservers	Yes
-P protocol	The protocol to allow.  Condition: Applies when specifying a CIDR subnet as the source.  Valid Values: tcp   udp   icmp  Example: -P tcp	Yes
-p port_range	For the TCP or UDP protocols, this specifies the range of ports to allow. You an specify a single integer or a range (min-max). Condition: Applies when specifying a CIDR subnet as the source. Example: -p 80	Yes
-t icmp_type_code	For the ICMP protocol, the ICMP type and code must be specified. This must be specified in the format type:code where both are integers. Type, code, or both can be specified as -1, which is a wildcard.  Condition: Applies when specifying a CIDR subnet as the source. Example: -t 2:5	Yes
-u source_group_user	The owner of a group specified using -o. If this is not specified, all groups will refer to the current user. If specified more than once, there must be exactly one -u per -o and each user will be mapped to the corresponding group.  Example: -u 495219933132	No
The network source from which traffic will be authorized specified as a security Group. See the description of the -u option for group owner information.  Example: -o headoffice		No
-s source_subnet	The network source from which traffic is to be authorized specified as a CIDR subnet range. Example: -s 205.192.8.45/24	No

#### **Output**

Amazon EC2 returns a table that contains the following information:

- Output type identifier ("GROUP", "PERMISSION").
- Group name. Currently, this will report an empty string.
- Type of rule. Currently, only ALLOW rules are supported.
- Protocol to allow.
- Start of port range.
- End of port range.
- FROM
- Source.

Amazon EC2 displays errors on stderr.

### **Example**

This example grants TCP port 80 access from the 205.192.0.0/16 address range to the websrv security group:

```
$ ec2-authorize websrv -P tcp -p 80 -s 205.192.0.0/16
GROUP websrv ""
PERMISSION websrv ALLOWS tcp 80 80 FROM CIDR 205.192.0.0/16
```

### **Related Topics**

- <u>AuthorizeSecurityGroupIngress</u>
- ec2-add-group
- ec2-describe-groups
- ec2-delete-group
- <u>ec2-revoke</u>

# ec2-confirm-product-instance

### **Syntax**

ec2-confirm-product-instance product\_code -i instance\_id

### **Description**

Returns a boolean indicating whether the specified product code is attached to the specified instance. If it is attached, It returns true. Otherwise, it returns false.

This command can only be executed by the AMI owner. This is useful when an AMI owner is providing support and wants to verify whether a user's instance is eligible.

### **Options**

Option	Description	Required
instance_id	Instance identifier that was generated when the instance launched. Example: i-10a64379	Yes
product_code	The product code. Example: 774F4FF8	Yes

#### Output

Amazon EC2 returns a table that contains the following information:

- Product code.
- Instance ID.
- Boolean indicating if the product code is attached to the instance.
- The instance owner's account ID (if the product code is attached).

Amazon EC2 displays errors on stderr.

### **Example**

This example confirms whether the 774F4FF8 product code is attached to the i-10a64379 instance:

**c2-confirm-product-instance** 774F4FF8 -i i-10a64379 true

## **Related Topics**

- <u>DescribeInstances</u>
- ec2-modify-image-attribute

# ec2-delete-group

### **Syntax**

ec2-delete-groupgroup

### **Description**

Deletes the specified security group.



If you attempt to delete a security group that contains instances, a fault is returned.

If you attempt to delete a security group that is referenced by another security group, a fault is returned. For example, if security group B has a rule that allows access from security group A, security group A cannot be deleted until the allow rule is removed.

# **Options**

Option	Description	Required
group	Name of the security group. Example: webservers	Yes

### Output

Amazon EC2 returns a table that contains the following information:

- Output type identifier ("GROUP").
- Name of the deleted security group.

Amazon EC2 displays errors on stderr.

### **Example**

This command deletes the *websrv* security group:

\$ ec2-delete-group websrv
GROUP websrv

### **Related Topics**

- <u>DeleteSecurityGroup</u>
- ec2-add-group
- ec2-describe-groups
- ec2-authorize
- <u>ec2-revoke</u>

# ec2-delete-keypair

### **Syntax**

ec2-delete-keypair key\_pair

# Description

Deletes the specifed key pair, by removing the public key from Amazon EC2

# **Options**

Option	Description	Required
key_pair	Name of the key pair. Example: primary_keypair	Yes

### Output

Amazon EC2 returns a table that contains the following information:

- Output type identifier ("KEYPAIR").
- Name of the deleted key pair.
- Private key fingerprint.

Amazon EC2 displays errors on stderr.

### **Example**

This example deletes the *gsg-keypair* key pair:

\$ ec2-delete-keypair gsg-keypair
KEYPAIR gsg-keypair

# **Related Topics**

- <u>DeleteKeypair</u>
- <u>ec2-add-keypair</u>
- ec2-describe-keypairs

# ec2-deregister

### **Syntax**

ec2-deregister ami\_id

### **Description**

Deregisters the specified AMI. Once deregistered, the AMI cannot be used to launch new instances.



This command does not delete the AMI from Amazon S3.

# **Options**

Option	Description	Required
ami_id	AMI identifier. Example: ami-4fa54026	Yes

### Output

Amazon EC2 returns a table that contains the following information:

- A record type identifier ("IMAGE")
- The image identifier that was deregistered

Amazon EC2 displays errors on stderr.

### **Example**

This example deregisters the *ami-4fa54026* AMI:

\$ ec2-deregister ami-4fa54026
IMAGE ami-4fa54026

# **Related Topics**

- <u>DeregisterImage</u>
- <u>ec2-register</u>
- ec2-describe-images

# ec2-describe-groups

### **Syntax**

ec2-describe-groups [group ...]

### **Description**

Describes the current state of each specified security group. If no security groups are explicitly listed, Amazon EC2 displays all security groups owned by the current user.

# **Options**

Option	Description	Required
group	Name of the security group. Example: webservers	Yes

#### Output

Amazon EC2 returns a table that contains the following information:

- Output type identifier ("GROUP", "PERMISSION").
- User ID of security group owner.
- Security group name.
- Description of the security group.
- Firewall rule.

Amazon EC2 displays errors on stderr.

### **Example**

This example displays the state of the websrv security group:

\$ ec2-describe-groups websrv
GROUP 495219933132 websrv Web servers
PERMISSION 495219933132 websrv ALLOWS tcp 80 80 FROM CIDR 0.0.0.0/0

### **Related Topics**

- <u>DescribeSecurityGroups</u>
- ec2-add-group
- ec2-delete-group
- ec2-authorize
- <u>ec2-revoke</u>

# ec2-describe-image-attribute

### **Syntax**

ec2-describe-image-attribute ami\_id ( -l | -p )

# Description

Describes an attribute for the specified AMI.

# **Options**

Option	Description	Required
ami_id	AMI identifier. Example: ami-4fa54026	Yes
-1	Describes the launchPermission attribute. Example: -1	Choice
-p	Describes the productCodes attribute. Example: -p	Choice

#### Output

Amazon EC2 returns a table that contains the following information:

- Attribute type identifier
- ID of the AMI.
- Attribute value type or attribute list item value type.
- Attribute or attribute list item value.

Amazon EC2 displays errors on stderr.

### **Examples**

This example lists the launch permissions for the ami-2bb65342 AMI:

```
$ ec2-describe-image-attribute ami-2bb65342 -l
launchPermission ami-2bb65342 group all
launchPermission ami-2bb65342 userId 495219933132
```

This example lists the product code for the ami-2bb65342 AMI:

```
$ ec2-describe-image-attribute ami-2bb65342 -p
productCodes ami-2bb65342 productCode 774F4FF8
```

### **Related Topics**

- <u>DescribeImageAttribute</u>
- ec2-modify-image-attribute
- ec2-reset-image-attribute
- Sharing AMIs

# ec2-describe-images

### **Syntax**

ec2-describe-images[ami\_id ...] [-a] [-o owner ...] [-x user\_id]

#### **Description**

Returns information about AMIs available to the user. This includes public AMIs available for any user to launch, private AMIs owned by the user making the request, and private AMIs owned by other users for which the user has explicit launch permissions.

The list of AMIs returned can be modified by specifying AMI IDs, AMI owners, or users with launch permissions. If no options are specified, Amazon EC2 returns all AMIs for which the user has launch permissions.

If you specify one or more AMI IDs, only AMIs that have the specified IDs are returned. If you specify an invalid AMI ID, a fault is returned. If you specify an AMI ID for which you do not have access, it will not be included in the results.

If you specify one or more AMI owners, only AMIs from the specified owners and for which you have access are returned. The results can include the account IDs of the specified owners, amazon for AMIs owned by Amazon or self for AMIs that you own.

If you specify a list of executable users, only users that have launch permissions for the AMIs are returned. You can specify account IDs (if you own the AMI(s)), self for AMIs for which you own or have explicit permissions, or all for public AMIs.



If you do not specify any optional parameters, Amazon EC2 returns images you own or images for which you have explicit access. Public images are not returned.

# **Options**

Option	Description	Required
-a	Returns AMIs that the user owns and for which the user has execution permissions.  Example: -a	No
-0 owner	Returns AMIs owned by the specified owner. Multiple owners can be specified. Owners are specified with AWS user account ID, without dashes. The IDs amazon, self, and explicit can be used to include AMIs owned by Amazon, AMIs owned by the user, and AMIs for which the user has explicit launch permissions, respectively.  Example: -o 495219933132	No
-x user_id	Returns AMIs for which the specified user has explicit launch permissions. The user ID can be a user's account ID, self to return AMIs for which the sender of the request has explicit launch permissions, or all to return AMIs with public launch permissions.  Example: -x self	No

#### **Output**

Amazon EC2 returns a table that contains the following information:

- A record type identifier ("IMAGE")
- Image identifier
- Manifest location
- User identifier of the user that registered the image
- Image status
- Image visability (public or private)
- Product codes, if any, that are attached to the instance

Amazon EC2 displays errors on stderr.

#### **Example**

This example describes the ami-78a54011 AMI:

```
$ ec2-describe-images ami-78a54011
IMAGE ami-78a54011 powerdns/image.manifest.xml 495219933132 available private 774F4FF8
```

This example describes the Amazon 64-bit AMI::

```
$ ec2-describe-images -o amazon | grep x86_64
IMAGE ami-78a54034 ec2-public-images/fedora-core6-base-x86_64.manifest.xml
amazon available public
```

# **Related Topics**

- <u>DescribeImages</u>
- <u>ec2-register</u>
- <u>ec2-deregister</u>

# ec2-describe-instances

### **Syntax**

ec2-describe-instances [instance\_id ...]

#### **Description**

Describes the current state of the specified instance(s). If you do not specify instances, all your instances are included in the output.

# **Options**

Option	Description	Required
	Instance identifier that was generated when the instance launched. Example: r-15a4417c	No

#### **Output**

Amazon EC2 returns a table that contains the following information:

- Output type identifier ("RESERVATION", "INSTANCE")
- Instance ID for each running instance.
- AMI ID of the image on which the instance is based.
- Public DNS name associated with the instance. This is only present for instances in the running state.
- Private DNS name associated with the instance. This is only present for instances in the running state, but is not valid for instances using legacy direct addressing.
- Instance state.
- Key name. If a key was associated with the instance at launch, its name will appear.
- AMI launch index. For more information, see <u>Instance Metadata</u>.
- Product codes attached to the instance.
- Instance type. The type of the instance (e.g., m1.small, m1.large, m1.xlarge).
- Instance launch time. The time the instance launched.

Amazon EC2 displays errors on stderr.

#### **Example**

This example describes the current state of the instances (currently one) owned by this user:

```
$ ec2-describe-instances

RESERVATION r-15a4417c 495219933132

INSTANCE i-3ea74257 ami-6ba54002 ec2-72-44-33-4.compute-1.amazonaws.com domU-12-31-33-00-6

INSTANCE i-31a74258 ami-6ba54002 ec2-72-44-34-23.compute-1.amazonaws.com domU-12-31-33-00-
```

# **Related Topics**

- <u>DescribeInstances</u>
- ec2-run-instances
- ec2-terminate-instances

# ec2-describe-keypairs

### **Syntax**

ec2-describe-keypairs [key\_id ...]

#### **Description**

Describes the current state of each specified key. If no keys are specified, all keys owned by the current user are included in the output.

# **Options**

Option	Description	Required
key_id	Names of one or more keys. Example: gsg-keypair	No

#### Output

Amazon EC2 returns a table that contains the following information:

- A output type identifier ("KEYPAIR")
- Key pair identifier
- Private key fingerprint

Amazon EC2 displays errors on stderr.

#### **Example**

This example describes the state of the current keys:

```
$ ec2-describe-keypairs gsg-keypair
KEYPAIR gsg-keypair 1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:f1:6f
```

# **Related Topics**

- <u>DescribeKeypairs</u>
- <u>ec2-add-keypair</u>
- <u>ec2-delete-keypair</u>

# ec2-fingerprint-key

### **Syntax**

ec2-fingerprint-key keyfile

#### **Description**

Computes and displays the fingerprint for a private key produced by Amazon EC2. .

This operation is performed entirely on the client-side. Network access is not required.

# **Options**

Option	Description	Required
	The path to a file containing an unencrypted PEM-encoded PKCS#8 private key. Example: mykey.pem	Yes

#### Output

A key fingerprint. This is formatted as a hash digest with each octet separated by a colon.

Amazon EC2 displays errors on stderr.

#### **Example**

This example computes and displays the fingerprint for the mykey.pem private key:

```
$ ec2-fingerprint-key mykey.pem
1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:f1:6f
```

# **Related Topics**

- <u>ec2-add-keypair</u>
- ec2-describe-keypairs

# ec2-get-console-output

### **Syntax**

ec2-get-console-output instance\_id [-r]

# **Description**

Retrieves the console output for specified instance, if available, and displays it to stdout.

# **Options**

Option	Description	Required
instance_id	Instance identifier generated when the instance lauched. Example: i-10a64379	Yes
-r	Raw output. Do not escape the output to facilitate reading.	No

#### Output

- A timestamp indicating the time of the last update.
- The instance console output. By default the ^ESC character is escaped and duplicate new-lines are removed to facilitate reading.

Amazon EC2 displays errors on stderr.

#### **Example**

This example retrieves the console output for the i-10a64379 instance

```
$ ec2-get-console-output i-10a64379
2007-01-03 12:00:00
Linux version 2.6.16-xenU (builder@patchbat.amazonsa) (gcc version 4.0.1 20050727 (Red Hat BIOS-provided physical RAM map:
Xen: 00000000000000000 - 000000006a400000 (usable)
980MB HIGHMEM available.
727MB LOWMEM available.
NX (Execute Disable) protection: active
IRQ lockup detection disabled
Built 1 zonelists
Kernel command line: root=/dev/sda1 ro 4
Enabling fast FPU save and restore... done.
```

# ec2-modify-image-attribute

### **Syntax**

```
ec2-modify-image-attribute ami_id -l (-a item_value | -r item_value)
ec2-modify-image-attribute ami_id -p product_code [-p product_code ...]
```

# **Description**

Modifies an attribute for the specified AMI.

#### **Attributes**

Attribute Name	Type	Description
launchPermission	List	Controls who has permission to launch the AMI. You can grant launch permissions by adding user IDs or make the AMI public by adding the all group. To learn more about sharing AMIs see <a href="Sharing AMIs">Sharing AMIs</a> .
		Note
		If another user launches your AMI there is no mechanism to prevent that user from rebundling the image and registering it as a new AMI.
productCodes	List	Associates a product code with an AMI. This allows a developer to charge a user for using the AMI.  Note
		The user must be signed up for the product before they can launch the AMI.
		The product code attribute is a write-once attribute. After a product code is set for an AMI it can not be altered or removed. AMIs are limited to one product code.

# **Options**

Option	Description	Required
-1	Modifies the launchPermission property. Example: -1	
-a item_value  -r item_value	Adds or removes an attribute item. The type of the item is inferred from the item value. For launchPermission there are two item types:  • group: The only group currently supported is the all group. Adding this group sets public launch permissions for the AMI.  • userId: The user ID is an AWS account ID, without dashes. Adding user IDs grants explicit launch permissions for the AMI.  Example: -a all	Yes
-p product_code	Sets the productCodes property. Example: -p 774F4FF8	Yes

#### **Output**

Amazon EC2 returns a table that contains the following information:

- Attribute type identifier.
- ID of the AMI on which attributes are being modified.
- Action performed on the attribute.
- Attribute or attribute list item value type.
- Attribute or attribute list item value.

Amazon EC2 displays errors on stderr.

#### **Examples**

The following example modifies launch permission attributes for the ami-2bb65342 AMI:

```
$ ec2-modify-image-attribute ami-2bb65342 -l -a 495219933132
launchPermission ami-2bb65342 ADD userId 495219933132
```

The following example adds the 774F4FF8 product code to the ami-2bb65342 AMI:

```
$ ec2-modify-image-attribute ami-2bb65342 -p 774F4FF8
productCodes ami-2bb65342 productCode 774F4FF8
```

### **Related Topics**

- ModifyImageAttribute
- ec2-reset-image-attribute
- ec2-describe-image-attribute
- Sharing AMIs

# ec2-reboot-instances

# **Syntax**

ec2-reboot-instances instance\_id [instance\_id ...]

# **Description**

Reboots one or more specified instances. You must specify at least one instance ID.

# **Options**

Option	Description	Required
instance_id	Instance identifier that was assigned to the instance at launch. Example: i-3ea74257	Yes

# Output

This command displays no output on success.

Amazon EC2 displays errors on stderr.

# Example

This example reboots the i-3ea74257 instance.

\$ ec2-reboot-instances i-3ea74257

# ec2-register

### **Syntax**

ec2-register manifest

### **Description**

Registers the Amazon Machine Image (AMI) specified in the manifest file and generates a new Amazon Machine Image (AMI) ID.

# **Options**

Option	Description	Required
manifest	Location and filename of the manifest file. The manifest file must be located in Amazon S3 and must be in the bucket/object form.  Example: mybucket/image.manifest.xml	Yes

# Output

Amazon EC2 assigns and returns an AMI ID.

Amazon EC2 displays errors on stderr.

#### **Example**

This example registers the AMI specified in the image.manifest.xml manifest file:

\$ ec2-register mybucket/image.manifest.xml
IMAGE ami-78a54011

# **Related Topics**

- RegisterImage
- <u>ec2-deregister</u>
- ec2-describe-images

# ec2-reset-image-attribute

# **Syntax**

ec2-reset-image-attribute ami\_id -l

# **Description**

Resets an attribute for the specified AMI.

The productCodes attribute cannot be reset.

# **Options**

Option	Description	Required
ami_id	The identifier that was assigned to the AMI when it was registered. Example: r-15a4417c	Yes
-1	Resets the launchPermission attribute. Example: -1	Yes

#### Output

Amazon EC2 returns a table that contains the following information:

- Attribute type identifier
- ID of the AMI on which the attribute is being reset
- Action identifier ("RESET")

Amazon EC2 displays errors on stderr.

### **Example**

This example resets the lauchPermission attribute:

\$ ec2-reset-image-attribute ami-6ba54002 -l
launchPermission ami-6ba54002 RESET

### **Related Topics**

- ResetImageAttribute
- ec2-modify-image-attribute
- ec2-describe-image-attribute
- Sharing AMIs

# ec2-revoke

#### **Syntax**

ec2-revoke group [-P protoco1] (-p port\_range | -t icmp\_type\_code) [-u
source\_group\_user ...] [-o source\_group ...] [-s source\_subnet ...]

### **Description**

Revokes a rule from the security group named GROUP. To identify the rule to be removed you must provide exactly the same set of options used to <u>create that rule</u>.

# **Options**

Option	Description	Required
group	The group to which this rule will apply. Example: webservers	Yes
-P protocol	The protocol to revoke.  Condition: Applies when specifying a CIDR subnet as the source.  Valid Values: tcp   udp   icmp  Example: -P tcp	Yes
-p port_range	For the TCP or UDP protocols, this specifies the range of ports to revoke. You an specify a single integer or a range (min-max). Condition: Applies when specifying a CIDR subnet as the source. Example: -p 80	Yes
-t icmp_type_code	For the ICMP protocol, you must specify the ICMP type and code. Use the type:code format, where both are integers. To specify a wildcard for either or both, enter -1.  Condition: Applies when specifying a CIDR subnet as the source. Example: -t 2:5	Yes
-u source_group_user	The owner of a group specified using -o. If this is not specified, all groups will refer to the current user. If specified more than once, there must be exactly one -u per -o and each user will be mapped to the corresponding group.  Example: -u 495219933132	No
-o source_group	The network source from which traffic will be revoked specified as a security Group. See the description of the -u option for group owner information.  Example: -o headoffice	No
-s source_subnet	The network source from which traffic is to be revoked specified as a CIDR subnet range.  Example: -s 205.192.8.45/24	No

#### Output

A table containing the following information is returned:

- Output type identifier ("GROUP", "PERMISSION").
- Group name. Currently, this will report an empty string.
- Type of rule. Currently, only ALLOW rules are supported.
- Protocol to allow.
- Start of port range.
- End of port range.
- FROM
- Source.

Amazon EC2 displays errors on stderr.

#### **Example**

This example revokes TCP port 80 access from the 205.192.0.0/16 address range for the websrv security group:

```
$ ec2-revoke websrv -P tcp -p 80 -s 205.192.0.0/16
GROUP websrv ""
PERMISSION websrv ALLOWS tcp 80 80 FROM CIDR 205.192.0.0/16
```

### **Related Topics**

- RevokeSecurityGroupIngress
- ec2-add-group
- ec2-describe-groups
- ec2-delete-group
- ec2-authorize

# ec2-run-instances

#### **Syntax**

ec2-run-instances ami\_id [-n instance\_count] [-g group [-g group ...]] [-k
keyname] [-d user\_data | -f user\_data\_file] [ --addressing addressing\_type] [
--instance-type instance\_type]

#### **Description**

Launches one or more instances of the specified AMI. Optional parameters include:

- A security group. New instances launch in this group. If no group is specified instances are launched in the default group.
- A key pair name. The public key associated with this key pair name are available to the instances at boot time.
- User data. This data is available to the launched instances. For more information, see <a href="Instance Metadata">Instance Metadata</a>.
- Addressing type. This specifies whether the instance has a NATted address.
   See <u>Instance Addressing and Network Security</u> for more information on instance addressing.
- This specifies the instance type.
   Options include m1.small, m1.large, and m1.xlarge.
   The default value is m1.small.
   For more information on instance types, see <u>Selecting Instance Types</u>

If the AMI has a product code attached for which the user has not subscribed, the **ec2-run-instances** call will fail.

# **Options**

Option	Description	Required
-n instance_count	The number of instances to launch. If Amazon EC2 cannot launch the specified number of instances, no instances will launch. If this is specified as a range (min-max), Amazon EC2 will try to launch the maximum number, but no fewer than the minimum number. Default: 1  Example: -n 5	No
-g group	The security group(s) within which to launch the instances. This determines the ingress firewall rules applied to the instances. If you specify more than one group, the security policy will be the union of the security policies of the specified groups.  Default: The default group.  Example: -g fooGroup	No
-k keyname	The key pair to make available to these instances at boot. Example: -k fooKeyPair	No
-d user_data	Data to make available to the instances. This data is read from the command line of the USER_DATA argument. If you want the data to be read from a file, see the -f option.  Example: -d "my user data"	No
-f user_data_file	Data to make available to these instances. The data is read from the file specified by FILE_NAME. To specify user data on the command line, use the -d option.  Example: -f data.zip	No
addressing addressing_type	The address type with which the instance launches.  Valid Value: public  Note  To support the early beta legacy networking scheme, direct is supported for early beta users.  For more information on instance addressing, see  Instance Addressing.  Example:addressing public	No
instance-type instance_type	The type of instance to launch. For more information, see <u>Selecting Instance Types</u> .  Example:instance-type m1.small	No

#### **Output**

Amazon EC2 returns a table that contains the following information:

- Output type identifier ("INSTANCE")
- Instance ID which uniquely identifies each running instance.
- AMI ID of the image on which the instance(s) are based.
- DNS name associated with the instance (only present for instances in the running state).
- Instance state. This is usually *pending*, which indicates that the instance(s) are preparing to launch.
- Key name. If a key was associated with the instance at launch its name is displayed
- AMI launch index. For more information, see <u>Instance Metadata</u>.
- Instance type. For more information on instance types, see <u>Selecting Instance Types</u>.
- Instance launch time. Specifies when the instance launched.

Amazon EC2 displays errors on stderr.

#### **Example**

This example launches five instances of the ami-6ba54002 AMI:

```
$ ec2-run-instances ami-6ba54002 -n 5
RESERVATION r-0ea54067 495219933132 default
INSTANCE i-3ea74257 ami-6ba54002 pending 0 m1.small 2007-07-11T16:40:44-
INSTANCE i-31a74258 ami-6ba54002 pending 1 m1.small 2007-07-11T16:40:44-
INSTANCE i-31a74259 ami-6ba54002 pending 2 m1.small 2007-07-11T16:40:44-
INSTANCE i-31a7425a ami-6ba54002 pending 3 m1.small 2007-07-11T16:40:44-
INSTANCE i-31a7425b ami-6ba54002 pending 4 m1.small 2007-07-11T16:40:44-
INSTANCE i-31a7425c ami-6ba54002 pending 5 m1.small 2007-07-11T16:40:44-
```

### **Related Topics**

- RunInstances
- ec2-terminate-instances
- ec2-describe-instances
- <u>ec2-add-keypair</u>
- Instance Metadata

# ec2-terminate-instances

### **Syntax**

ec2-terminate-instances instance\_id [instance\_id ...]

# Description

Terminates the specified instances.

# **Options**

Option	Description	Required
instance_id	ID of the instance that was assigned at lauch. Example: i-3ea74257	Yes

#### Output

Amazon EC2 returns a table that contains the following information:

- Output type identifier ("INSTANCE")
- The instance ID of the instance being terminated.
- The state of the instance prior to being terminated.
- The new state of the instance.

Amazon EC2 displays errors on stderr.

### **Example**

This example terminates the i-3ea74257 instance:

\$ ec2-terminate-instances i-3ea74257
INSTANCE i-3ea74257 running shutting-down

# **Related Topics**

- <u>TerminateInstances</u>
- ec2-run-instances
- ec2-describe-instances

# **Technical FAQ**

#### **Topics**

- General Information
- Operation Information
- Instance Types and Architectures
- <u>IP Information</u>
- Monitoring, Errors, and Unexpected Behavior
- Error Messages
- Paid AMIs
- Miscellaneous

This section contains answers to commonly asked questions.

#### **General Information**

How many instances can I launch?

Each user has a concurrent running instance limit. For new users, this limit is 20. If you need more than 20 instances, please contact aws@amazon.com.

How do I sign a request?

Information on signing SOAP requests is provided in <u>Request Authentication</u>. Information on signing Query requests is provided in <u>Query API Authentication</u>

What username do I use for the various Amazon EC2 tools?

When you sign up with Amazon Web Services, you are provided an AWS Account ID. This is your username. For more information, refer to the *Amazon Elastic Compute Cloud Getting Started Guide*.

Why do my instances take so long to start?

Amazon EC2 must move the images around the network before they can be launched. For big images and/or congested networks, this can take several minutes. To improve performance, images are cached. As you launch your images more frequently, it should be less noticeable.

How durable are the instance stores?

Instance stores appear to an instance as a local disk. They will survive intentional and unintentional reboots of the instance unless the instance terminates or the underlying drive fails.

You should always backup or replicate important data.

What happens to my running instances if the machines on which they are running go down?

The instances will terminate and will need to be relaunched. The data on the instances' hard drives will be lost.

Always replicate important data or store it in Amazon S3.

Can I use my own kernel?

Not at present.

Can I get a bigger/smaller/differently optimized virtual machine?

Yes. For more information, see **Selecting Instance Types**.

*Is there a REST interface to Amazon EC2?* 

Not at present. You can use the SOAP API, Query API, or the command-line tools.

How does Amazon EC2 handle load balancing?

With a service as flexible as Amazon EC2, you can use many types of load balancing systems. The load balancing instances can forward traffic to other systems. There are several open source solutions that are in wide use.

Does Amazon perform system maintenance?

Yes. Periodically, Amazon might perform maintenance that requires a reboot of your system. Make sure your instances can recover and restart after being rebooted.

# **Operation Information**

How do I handle time synchronization between instances?

You can set up NTP (Network Time Protocol). For more information, go to <a href="http://www.ntp.org/">http://www.ntp.org/</a>. NTP is particularly important if you plan on using any Amazon web services (such as Amazon S3 or Amazon EC2) from within an instance, since requests to these services must be timestamped.

*Is there a method for an instance to discover its own instance ID?* 

From within your instance you can use REST-like queries to http://169.254.169.254/2007-08-29/ to retrieve various instance-specific metadata, including the instance ID. For more information, see <a href="Instance Instance">Instance Instance Insta

Can I pass arbitrary configuration values to an instance at launch time?

Yes, although the size of the data is limited to 16K. For more information, see Instance Metadata.

Is there a way to run a script on instance termination?

Not with any reliability. Amazon EC2 tries to shut an instance down cleanly (running system shutdown scripts), but there is only a short time available. In some cases (e.g., hardware failure), this does not happen.

Since there is no way to ensure shutdown scripts run, have a strategy to deal with abnormal terminations.

How can I allow other people to launch my AMIs?

You can allow other users to launch your AMIs by modifying the AMI's launchPermission attribute. You can grant public launch permissions or explicit permissions to specific users.

To grant public launch permissions:

To grant a specific user launch permissions:

```
PROMPT> ec2matt ami-2bb65342 -t launchPermission -a -i userId=49521993313
```

To clear additional launch permissions for an AMI:

```
PROMPT> ec2ratt ami-2bb65342 -t launchPermission
```

Why do I need to reregister a rebundled AMI? Can I keep the same AMI ID?

An AMI ID is associated with the physical bits in an image. To protect users from images being modified, we require you to reregister AMIs after rebundling.

*Can I pass JVM properties to the command line tools?* 

Yes. By setting the environment variable EC2\_JVM\_ARGS, you can pass arbitrary JVM properties to the command line tools.

Can I use a proxy with the command line tools?

Yes. By passing in JVM properties through the EC2\_JVM\_ARGS environment variable, you can specify proxy settings for the command line tools. For example, in Linux:

The following properties are supported for configuring a proxy:

Setting	Description
https.proxyHost	HTTPS proxy host
https.proxyPort	HTTPS proxy port
http.proxyHost	HTTPS proxy host
http.proxyPort	HTTPS proxy port

http.proxyRealm	Proxy realm (https and http)
http.proxyUser	Proxy username (https and http)
http.proxyPass	Proxy password (https and http)



https.proxyHost should be used when EC2\_URL points to an https host, and http.proxyHost when EC2\_URL points to an http host.

# **Instance Types and Architectures**

What happened to the original instance type?

The original instance type is still available. It is called the small instance (m1.small) and it has the same technical specifications.

Will the original instance type be retired soon?

There are no plans to retire the original instance type.

If I do not specify an instance type at launch, what type of instance will I get?

You will get a small instance instance type which is equivalent to the original Amazon EC2 instance type.

Does my instance limit apply to all instance types or is there a separate limit for each type?

The instance limit applies to the sum of all instances, regardless of type. There is no separate instance limit per type.

Can I mix instance types, or do I have to use the same type for all of my instances?

You can launch any combination of instance types. Choose the instance types that have the most appropriate memory, CPU, and storage for each function within your application.

Which instance types are 32-bit and which are 64-bit?

The small Instance (m1.small) is 32-bit. The large (m1.large) and extra large (m1.xlarge) instances are 64-bit.

Can I launch any AMI on any type of instance?

No. You must use 64-bit AMIs on large (m1.large) and extra large (m1.xlarge) instances and 32-bit AMIs on small Instances (m1.small).

Do I have to do anything special to bundle the large or extra large instances?

Make sure to use the latest AMI Tools.

Can I build an AMI that works on both 32-bit and 64-bit instances?

No, an AMI is either a 32-bit AMI or a 64-bit.

Can I run 32- bit applications on 64-bit AMIs?

You can run a 32-bit application on a 64-bit host if the Linux kernel is compiled with IA32 emulation and the correct 32-bit libraries are available.

By default, the Amazon DomU Kernel has IA32 emulation enabled and there are many public AMIs that include pre-installed 32-bit libraries. If the library you require is not included with the AMI, you can install it using standard tools (e.g., yum).

How fast is the disk?

The large and extra large instances have higher and more consistent I/O performance than the original (small) instance.



The first write to any given block of the disk will be slower than subsequent writes. For more information, see <u>Disk Performance Optimization</u>

Can I RAID the spindles exposed on large and extra large instances?

Yes, you can use software RAID on top of the exposed spindles.



The initial RAID setup might take a long time. For more information, see <u>Disk</u> <u>Performance Optimization</u>

Can I launch the large or extra large instance in direct addressed space?

No, the large and extra large instances are only available as NAT addressed instances. For more information, see <a href="Instance Addressing and Network Security">Instance Addressing and Network Security</a>.

#### **IP Information**

How do I host a public domain if I have to DHCP an IP address?

You can use a dynamic DNS service, such as **DynDNS** or **ZoneEdit**.

Why can't I connect to my instances public IP address from another instance?

There is a known limitation that prevents direct-addressed instances from contacting NAT-addressed instances through their public IP address. You should use the internal IP/DNS name when communicating between Amazon EC2 instances. This will ensure the lowest latency, highest throughput, and lowest cost network route.

Why do I get an internal (RFC 1918) IP address when I look up a DNS name that I expect to map to my instance's external IP address?

Amazon EC2 instances cannot currently access other instances in the new NAT environment using their public NAT IP address. Instead, they must use the private address. Traffic originating from the Internet must use the public NAT IP address. Within Amazon EC2, DNS requests for the external DNS name of an instance will resolve to the internal IP address of the corresponding instance

Can I use IPv6?

Amazon EC2 presently supports the ICMP, TCP and UDP protocol families of IPv4-based networking.

Why is Amazon EC2 Using NAT?

Public IP space is a limited resource. Amazon EC2 is adopting NAT to assure that we are able to efficiently make use of our public internet addresses.

Furthermore, the new NAT networking will enable Amazon to deliver new features in the future. For example, some users do not want external addresses. This would allow for non-internet routable clusters, which will further preserve IPs and increase security for those not running public facing servers.

Can I use a static IP in my instances?

Not at present. Your image must be configured as a DHCP client and it will be assigned an IP address. Currently, all instances come with internet- addressable IP addresses. If you enable access through the firewall from the "world", you can address them from anywhere.

How does the instance know its public and private addresses?

From within the instance, issue the following HTTP queries:

To obtain the internal IP address:

```
curl http://169.254.169.254/2007-08-29//meta-data/local-ipv4
```

To obtain the public IP address:

```
curl http://169.254.169.254/2007-08-29//meta-data/public-ipv4
```

# Monitoring, Errors, and Unexpected Behavior

*How do I monitor my systems?* 

Amazon EC2 provides basic monitoring. You can use DescribeInstances to check whether an instance appears to be running. However, if you are using Amazon EC2 as your data center, you might want to set up for sophisticated monitoring on your instances, such as SNMP.

Why can't I "talk" to my instances?

There are a few common reasons for broken connectivity to your instance.

Amazon EC2 changes the state of your instance to running after your operating system starts booting. Depending on your AMI, there will be a delay before the instance is fully set up and functional.

If your instance has been running for several minutes, you verify you authorized the appropriate access to your host through the Amazon EC2 firewall. If you have launched your instances without specifying a security group, the default group is used. Permissions on the default group are very strict and disallow all access from the Internet and other groups. You will need to modify the permissions of your default group or set up a new group with appropriate permissions. For more information, see <a href="Network Security">Network Security</a>

If this doesn't solve your issue, make sure you authorized port 22 and try to open an SSH connection with verbose output. Use the man page for the exact syntax of your system, but the command is likely to be similar to ssh -vv root@[hostname]. This output is very useful if you are posting to the forum.

Why did my instance terminate immediately after launch?

Launch errors can be the result of an internal error during launch or a corrupt Amazon EC2 image. Internal errors are rare, as we actively test for and isolate suspect hosts. Consult the DescribeInstances operation for details on why your instance failed to launch.

#### Note

The ec2-describe-instances command line tool does not provide this information. Use the -v flag to read the detailed SOAP response and get detailed information.

You can also attempt to launch the image again. If this proves to be a persistent problem (especially with a shared image), post to the <u>AWS forums</u>.

I ran shutdown from within an ssh session, but my instance still shows up as running when I query it with DescribeInstances and I can't shell into it.

This is a "feature" of the shutdown command. If you issue shutdown without a - h (halt) flag, it shuts down the network and switches to single user mode. The instance is still running, but without a network.

You should always use shutdown -h when working inside an Amazon EC2 instance.

You can also shut the instance down using the TerminateInstances call (ec2-terminate) on the command line.

Why are my instances stuck in a pending state (or a shutting-down state)?

This situation is rare and might be the result of a software error or misconfiguration.

We actively monitor for this; please contact us if it occurs.

Why do I get an "AuthFailure: User is not AMI creator" error when I try to register an image?

Make sure that you are using the correct user ID and certificate to create and upload the image. You must use the same ID and certificate to register the image with Amazon EC2.

# **Error Messages**

Why do I get an "InsufficientInstanceCapacity" error when I try to launch an instance?

This error indicates that we do not currently have enough available capacity to service your request.

If you are requesting a large number of instances, there might not be enough server capacity to host them. You can try again later or specify a smaller number of instances.

Why do I get an "InstanceLimitExceeded" error when I try to launch an instance?

This error indicates you reached your concurrent running instance limit. For new users during the public beta, the limit is 20.

If you need additional capacity, please contact us at aws@amazon.com.

Why can't I retrieve my instance-specific data from within a running instance when querying http://169.254.169.254/2007-08-29/?

The Parameterized Launches feature is available to instances that were launched after the feature was released. If you launched your instance before this, the data will not be available. If you want to use this functionality, relaunch your instances.

If, after relaunching your instance, you still experience problems retrieving the data, check the following:

- Are you using the correct base URI (http://169.254.169.254/2007-08-29/)
- Are you using the correct URI for the data you're trying to retrieve? Depending on the data you're trying to retrieve, a trailing '/' might be required.
- Did you specify any launch data when launching your instances? If not you will get a HTTP error response (404) when trying to retrieve the user data.

#### Note

Instance metadata is always available, even if you do not specify it at instance launch.

Why do I get keep getting "Request has expired" errors?

To reduce the risk of replay attacks, our requests include a timestamp. This and the most important parts of the request are signed to ensure the message (including the timestamp) cannot be modified without detection.

If the difference between the timestamp in the request and the time on our servers is larger than 5 minutes, the request is too old (or too new) and an error is returned.

You need to ensure that your system clock is accurate and configured to use the correct timezone. For more information, go to NTP.

#### **Paid AMIs**



You can still share AMIs without charging. Public and paid AMIs can be listed in the Resource Center.

How can I determine if a particular AMI is a paid AMI?

By describing images (ec2dim) with the "-a" flag and looking for AMIs that have a product code. For example, if you run ec2dim -a, the result contains an AMI with the ID ami-bd9d78d4. This is our Demo Paid AMI with product code A79EC0DB.

How can I determine if a public AMI is paid?

By describing images (ec2dim). An AMI is a paid AMI if a product code is returned. Example: run ec2dim -a amazon, and the AMI ami-bd9d78d4 will be returned with a product code (A79EC0DB).

*Is there anything that prevents a paid AMI from being rebundled? How can this be restricted?* 

Paid AMIs are comparable to shared AMIs with regards to rebundling and trying to restrict rebundling. If you allow a user running the AMI to see all of its contents (e.g. by giving root access to the AMI), the user could rebundle these into their own AMI.

Why can't I query a particular AMI's attributes to see if the AMI is paid?

Only the owner of an AMI can query the AMI attributes. However, anyone can tell if an AMI is paid by describing images (ec2dim). An AMI is paid if a product code is returned. Example: run ec2dim -a amazon, and the AMI with ID ami-bd9d78d4 will be returned with a product code (A79EC0DB).

Who can use the confirm-product-instance command?

Only the owner of the AMI can use this command. Owners use this command with supported AMIs to determine if a supported instance with a given product

code attached is up and running.

Will the product code be inherited by the rebundled AMI?

If your customer uses AWS tools to rebundle the AMI, the product code associated with the AMI is inherited by the rebundled AMI. When launching the rebundled AMI the customer is still billed for usage based on your price.



This is a convenience feature and not a guarantee that the product code will always be attached to rebundled AMIs.

Note that the customer's workflow could bundle the AMI outside of Amazon EC2, or the customer could use modified versions of the AWS tools, preventing the product code from being inherited.

Will the kernel/ramdisk be inherited by the rebundled AMI?

If the AMI is rebundled on an instance that is running with a certain kernel/ramdisk, then the kernel/ramdisk will be inherited by the rebundled AMI.



This is a convenience feature and not a guarantee that the kernel/ramdisk will always be attached to rebundled AMIs.

Who can use the confirm-product-instance command?

If the AMI is rebundled on an instance that is running with a certain kernel/ramdisk, then the kernel/ramdisk will be inherited by the rebundled AMI.



This is a convenience feature and not a guarantee that the product code will always be attached to rebundled AMIs.

#### **Miscellaneous**

*Are there any special requirements to use FTP?* 

The File Transfer Protocol (FTP) has a PORT command by which a client sends its address back to the server. The server then connects to the client at that address to send the file data. If the client looks up its own internal address and sends this to the server, the connection will fail. In this specific case, there are two solutions to the problem. First, configure the client to send its public IP address. Second, the client can use "passive FTP" which makes connections only to the server, rather than from the server to the client. In general, applications which encode local addresses and port numbers in data sent to external servers might have problems with NAT. Care must always be taken to send the public address, rather than the internal one.

We recommend using passive mode unless it is not supported by the FTP server.

# **Glossary**

#### Amazon machine image (AMI)

An Amazon Machine Image (AMI) is an encrypted machine image stored in Amazon S3. It contains all the information necessary to boot instances of your software.

#### explicit launch permission

Launch permission granted to a specific user.

#### instance

Once an AMI has been launched, the resulting running system is referred to as an instance. All instances based on the same AMI start out identical and any information on them is lost when the instances are terminated or fail.

#### group

Also known as a security group, groups define firewall rules that can be shared among a group of instances that have similar security requirements. The group is specified at instance launch.

#### launch permission

AMI attribute allowing users to launch an AMI

#### public AMI

An AMI that all users have launch permissions for.

#### reservation

A collection of instances started as part of the same launch request.

#### shared AMI

AMIs that developers build and make available for other AWS developers to use.

### **Document Conventions**

This section establishes the common typographical and symbol use conventions for AWS technical publications.

# **Typographical Conventions**

This section describes common typographical use conventions.

Convention	Description/Example
Call-outs	A call-out is a number in the body text to give you a visual reference. The reference point is further discussion elsewhere.
	You can use this resource regularly.
Code in text	Inline code samples (including XML) and commands are identified with a special font.
	You can use the command java -version.
Code blocks	Blocks of sample code are set apart from the body and marked accordingly.
	<pre># ls -l /var/www/html/index.html -rw-rw-r 1 root root 1872 Jun 21 09:33 /var/www/html/index # date Wed Jun 21 09:33:42 EDT 2006</pre>
Emphasis	Unusual or important words and phrases are marked with a special font.
	You <i>must</i> sign up for an account before you can use the service.
Internal cross references	References to a section in the same document are marked.
	See <u>Document Conventions</u> .
Logical values, constants, and regular expressions, abstracta	A special font is used for expressions that are important to identify, but are not code.
	If the value is null, the returned response will be false.
Product and	Named AWS products and features are identified on first use.
feature names	Create an Amazon Machine Image (AMI).
Operations	In-text references to operations.
	Use the GetHITResponse operation.
Parameters	In-text references to parameters.
	The operation accepts the parameter <i>Account ID</i> .

Response elements	In-text references to responses.
cicilients	A container for one CollectionParent and one or more CollectionItems.
Technical publication references	References to other AWS publications. If the reference is hyperlinked, it is also underscored
	For detailed conceptual information, see the <i>Amazon Mechanical Turk Developer Guide</i> .
User entered values	A special font marks text that the user types.
	At the password prompt, type MyPassword.
User interface controls and labels	Denotes named items on the UI for easy identification.
	On the <b>File</b> menu, click <b>Properties</b> .
Variables	When you see this style, you must change the value of the content when you copy the text c sample to a command line.
	% ec2-register < <i>your-s3-bucket</i> >/image.manifest
	See also the symbol convention below.

# **Symbol Conventions**

This section describes the common use of symbols.

Convention	Symbol	Description/Example
Mutually exclusive parameters	(Parentheses   and   vertical   bars)	Within a code description, bar separators denote options from which one must be chosen.
		% data = hdfread (start   stride   edge)
Optional parameters	[square brackets]	Within a code description, square brackets denote completely optional commands or parameters.
XML variable text		% sed [-n, -quiet]
		Use square brackets in XML examples to differentiate them from tags.
		<customerid>[ID]</customerid>
Variables	<arrow brackets&gt;</arrow 	Within a code sample, arrow brackets denote a variable that must be replaced with a valid value.
		% ec2-register <your-s3-bucket>/image.manifest</your-s3-bucket>