Amazon EC2

Developer Guide

Amazon EC2: Developer Guide

Table of Contents

Introduction

Welcome to the Amazon Elastic Compute Cloud (Amazon EC2) Developers Guide.

This guide picks up where the Getting Started Guide ends and will provide you with the information necessary for creating more sophisticated AMIs, using advanced service features, and writing applications using Amazon EC2. This guide assumes you have worked through the Getting Started Guide, installed the command line and API tools as described, and have a general understanding of the service.

The chapters presented in the guide are as follows :

- *Creating and Bundling AMIs* walks you through the steps required to create the customized package of software that will execute on your host - essentially packaging your desired Operating System configuration.
- *Securing the Network* provides an overview of the distributed firewall and usage examples.
- *Using Instances* provides an overview of the Amazon EC2 instances and some tips for using them effectively.
- *Command Line Tool Reference* provides a comprehensive reference to the command line tools supplied by Amazon EC2.
- *Using the SOAP API* explains how to use the WS-SOAP interface to Amazon EC2.
- *API Reference* provides a comprehensive reference to the web services (SOAP) API exposed by Amazon EC2.
- *Technical FAQ* is a collection of interesting and commonly asked questions.

Creating and Bundling AMIs

Amazon EC2 allows you to set up and configure everything about your instances from your operating system up to your applications. An Amazon Machine Image (AMI) is simply a packaged-up copy of your root file system. An AMI contains all the necessary bits to set up and boot an Amazon EC2 instance. Your AMIs are your unit of deployment. You might have just one AMI or you might compose your system out of several building block AMIs (e.g., webservers, appservers, and databases).

This section details how to build your own AMIs and how to store them in Amazon S3. In addition, it covers booting existing images, and bundling those (after making any required changes). Re-bundling modified versions of existing AMIs provides an easy way to build up a range of different machine images from a pool of existing base images. Amazon EC2 provides base AMIs to get you started.

Creating an AMI

There are several techniques for creating an AMI offering 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 the section called “Starting with an Existing AMI”.

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 the section called “Creating via a Loopback File”.

Once the installation package has been built to your satisfaction it needs to be bundled and uploaded to Amazon S3 as described in the section called “Bundling an AMI”.

Starting with an Existing AMI

This is the quickest and easiest of the methods to get a new working AMI. Start with an existing public AMI or one of your own. You can then modify that as you see fit and subsequently create a new AMI with the **ec2-bundle-vol** utility, as decribed later in the section called “Bundling an AMI”.

Select an AMI

The first step is to locate an AMI that contains the packages and services that you require. This can be one of your own AMIs or one of the public AMIs provided by Amazon EC2. Use **ec2-describe-images** to get a list of available AMIs, as is shown below, then select one of the listed AMIs and note its AMI ID, e.g. ami-61a54008:

```
PROMPT>  ec2-describe-images
IMAGE ami-60a54009 ec2-public-images/base-fc4-apache.manifest 475219833042
available public
IMAGE ami-61a54028 <your-s3-bucket>/image.manifest 495219933132 available
private
IMAGE ami-61a54008 ec2-public-images/getting-started.manifest 475219833042
available public
IMAGE ami-6ea54007 ec2-public-images/base-fc3-mysql.manifest 475219833042
available public
```

Generate a Keypair

This step is only required if you've selected one of the public AMIs provided by Amazon EC2. A public/private keypair must be created to ensure that you, and only you, have access to the instances that you launch.

```
PROMPT>  ec2-create-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-----
MIICXAIBAAKBgQC2i/Sgs5BGGd4sunpYQfEkcprgzP9M/hnVJTc1j0nZBeIE2JBuLRSNoqkO7Gw8
nBcdNptaLedzqN8t78jGkX1TPWVAKJTfxRSvU/oViGJaRqIBar0Mpc/wC27kyzHezUNS5+mvONb3
4h/j2EZwDLY75Uxrpka0aN6OkvyIP5gYMQIDAQABAoGAOKH65tBOdjEYSHAh/LeYhGI5wnxWyCAd
C49cLXWix32XvUEircu2kKpiIIsgmT0jvqBuWe/b2noNo0a81z3TzzRYyLvn5J8mUlL6a8nsssQ3
xCHkGM+SE7ZzfBS5WUkbh5Exd3ZXKfCJvJW6auOzJ581JB5yUNbqixWzHuQGGAECQQDwq4LQoyb5
OVSpZwSy+GW/p0yRsqRp89ECNQ+hySGBjkSXBcbt75C+5ebo88/V2V4QOGGa0T0tMsMgKTJ8oukh
AkEAwiyoFM0Zwk0Os3rBZ8PyZoNW5e5SBwrEbLRv4JCaNiQme0ighsDr2bL/nGLI7p13g22+9REM
i/WAmsln50H9EQJASMun6tGepT2pFQBbFIM7y4egCmXdg0rDSoagLtB2eQh+SKvvquKOhp9lg8rT
b5yq7f8PztNBTN2Q1baAVeC04QJAGgN5kS/ZH5rLOWhcuNYbh3hZD/zZqG/c2ONjiaZVwqMdNK8K
MoNuFYBRllX1rWITPNxbFOHv2GBPlm0dKnJAwQJBAOgwjgLY3UpXFX9ZvG4RGEYgfui49Ffz10CH
5sSZpsFYn42E6a2NUJeL4hTzfbGTQ8iCIVjOXFH/9XLTDCNQEPM=
-----END RSA PRIVATE KEY-----
```

The resulting private key must be saved in a local file for later use. Create a file named `id_rsa-gsg-keypair` and paste into it all lines starting with the line "`-----BEGIN PRIVATE KEY-----`" and ending with the line "`-----END PRIVATE KEY-----`". Confirm that the file contents looks exactly as shown below.

```
-----BEGIN RSA PRIVATE KEY-----
MIICXAIBAAKBgQC2i/Sgs5BGGd4sunpYQfEkcprgzP9M/hnVJTc1j0nZBeIE2JBuLRSNoqkO7Gw8
nBcdNptaLedzqN8t78jGkX1TPWVAKJTfxRSvU/oViGJaRqIBar0Mpc/wC27kyzHezUNS5+mvONb3
4h/j2EZwDLY75Uxrpka0aN6OkvyIP5gYMQIDAQABAoGAOKH65tBOdjEYSHAh/LeYhGI5wnxWyCAd
C49cLXWix32XvUEircu2kKpiIIsgmT0jvqBuWe/b2noNo0a81z3TzzRYyLvn5J8mUlL6a8nsssQ3
xCHkGM+SE7ZzfBS5WUkbh5Exd3ZXKfCJvJW6auOzJ581JB5yUNbqixWzHuQGGAECQQDwq4LQoyb5
OVSpZwSy+GW/p0yRsqRp89ECNQ+hySGBjkSXBcbt75C+5ebo88/V2V4QOGGa0T0tMsMgKTJ8oukh
AkEAwiyoFM0Zwk0Os3rBZ8PyZoNW5e5SBwrEbLRv4JCaNiQme0ighsDr2bL/nGLI7p13g22+9REM
i/WAmsln50H9EQJASMun6tGepT2pFQBbFIM7y4egCmXdg0rDSoagLtB2eQh+SKvvquKOhp9lg8rT
b5yq7f8PztNBTN2Q1baAVeC04QJAGgN5kS/ZH5rLOWhcuNYbh3hZD/zZqG/c2ONjiaZVwqMdNK8K
MoNuFYBRllX1rWITPNxbFOHv2GBPlm0dKnJAwQJBAOgwjgLY3UpXFX9ZvG4RGEYgfui49Ffz10CH
5sSZpsFYn42E6a2NUJeL4hTzfbGTQ8iCIVjOXFH/9XLTDCNQEPM=
-----END RSA PRIVATE KEY-----
```

Launch an Instance

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

```
PROMPT>  ec2-run-instances ami-61a54008 -k gsg-keypair
INSTANCE        i-10a64379   ami-61a54008   EC2    pending   gsg-keypair
```

The instance ID in the second field of the output is a unique identifier for the instance and can be used subsequently to manipulate your instance, e.g. to terminate it.

> **Important**
> Once you launch an instance, you will be billed per hour for CPU time. Make sure you terminate any instances which you don't intend to leave running indefinitely.

It will take a few minutes for the instance to launch. You can follow its progress by running:

```
PROMPT>  ec2-describe-instances i-10a64379
RESERVATION     r-fea54097  495219933132   EC2
INSTANCE        i-10a64379  ami-61a54008   domU-
12-34-31-00-00-05.usma1.compute.amazonaws.com   EC2    running   gsg-keypair
```

When the status field reads "running", the instance has been created and has started booting. There may still be a short time before it is accessible over the network, however. The DNS name displayed in the sample output above will be different from that assigned to your instance. Make sure you use the appropriate one.

Authorize Network Access

In order to be able to reach the running instance from the Internet, you need to enable access for the ssh service which runs on port 22:

```
PROMPT>  ec2-authorize default -p 22
PERMISSION     default  ALLOWS  tcp     22      22      FROM    CIDR
0.0.0.0/0
```

Connect to the Instance

Now that you have a running instance, you can log in and modify it according to your requirements. If you launched a public Amazon EC2 AMI, you can use the following command to log in with your own private key:

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

Otherwise, use the plain ssh command and supply the appropriate password when prompted.

```
PROMPT>  ssh root@domU-12-34-31-00-00-05.usma1.compute.amazonaws.com
root@my-instance #
```

You now have complete control over the instance and may add, remove, modify or upgrade packages and files to suit your needs. Some of the basic configuration settings related to the Amazon EC2 enviroment, such as the network interface configuration and `/etc/fstab` contents, should only be changed with extreme care, to avoid making the AMI unbootable or inaccessible from the network once running.

Upload the Key and Certificate

The new AMI will be encrypted and signed to ensure that it can only be accessed by you and Amazon EC2. You therefore need to upload your Amazon EC2 private key and X.509 certificate to the running instance, for use in the AMI bundling process.

Assuming the private key and X.509 certificate are contained in files `pk-HKZYKTAIG2ECMXYIBH3HXV4ZBZQ55CLO.pem` and `cert-HKZYKTAIG2ECMXYIBH3HXV4ZBZQ55CLO.pem`, copy both of these files to your instance:

```
PROMPT> scp pk-HKZYKTAIG2ECMXYIBH3HXV4ZBZQ55CLO.pem cert-HKZYK-
TAIG2ECMXYIBH3HXV4ZBZQ55CLO.pem root@domU-
12-34-31-00-00-05.usma1.compute.amazonaws.com:/tmp
pk-HKZYKTAIG2ECMXYIBH3HXV4ZBZQ55CLO.pem                         100%  717
0.7KB/s   00:00 cert-HKZYKTAIG2ECMXYIBH3HXV4ZBZQ55CLO.pem
100%  685     0.7KB/s   00:00
```

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

You are now ready to proceed to the next step which involves bundling the volume and uploading the resulting AMI to Amazon S3. This is described in the section called “Bundling an AMI”.

Creating via a Loopback File

This method entails 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.

Create a File to Host the AMI

The **dd** utility can be used to create files of arbitrary sizes. In this case, 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 1GB. The command below creates a file of 1024*1MB=1GB.

```
# 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 generic **mkfs** utility that can be used to create a file system inside `my-image.fs`. Typical Linux installations default to `ext2` or `ext3` file systems. Create an `ext3` file system by issuing the following command:

```
# 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 via Loopback

The loopback module allows you to use a normal file as if it were a raw device. In this manner you get a file-system in a file. Mounting a file system image file via loopback presents it as part of the normal file system. You can then modify it using your favourite file management tools and utilities. Create a mount point in the file system where the image will be attached and then mount the file system image, as follows:

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

Prepare for the Installation

Before the operating system installation can proceed, some basic files have to be created and prepared on the newly created root file system.

Create `/dev`

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

```
MAKEDEV: mkdir: File exists
MAKEDEV: mkdir: File exists
```

Create `/etc`

Create an `/etc` directory:

```
# mkdir /mnt/ec2-fs/etc
```

Create `/mnt/ec2-fs/etc/fstab` and add the following entries to it:

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

Create `yum-xen.conf`

Create a temporary **yum** configuration file that will ensure all the required basic packages and utilities are installed. This configuration file can be created anywhere on your main file system, but for now we'll assume that you create it in your working directory. Just to clarify, it does not need to be created in the loopback file system. It is used only during installation of the loopback file system. Create `yum-xen.conf` with the following content:

```
[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 - $basearch - Base
mirrorlist=http://fedora.redhat.com/download/mirrors/fedora-core-$releasever
enabled=1

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

Mount `proc`

Due to a bug in the **groupadd** utility from the `shadow-utils` package (versions prior to 4.0.7-7), the new `proc` file system needs to be mounted by hand at this point.

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

Install the Operating System

At this stage all the basic directories and files have been created and you are ready to do the operating system installation. This process might take a while depending on the speed of the host and the network link to the repository.

```
# yum -c yum-xen.conf --installroot=/mnt/ec2-fs -y groupinstall Base
```

```
Setting up Group Process
Setting up repositories
base                      100% |=========================| 1.1 kB    00:00
updates-released          100% |=========================| 1.1 kB    00:00
comps.xml                 100% |=========================| 693 kB    00:00
comps.xml                 100% |=========================| 693 kB    00:00
Setting up repositories
Reading repository metadata in from local files
primary.xml.gz            100% |=========================| 824 kB    00:00
base      : ################################################## 2772/2772
Added 2772 new packages, deleted 0 old in 15.32 seconds
primary.xml.gz            100% |=========================| 824 kB    00:00
updates-re: ################################################## 2772/2772
Added 2772 new packages, deleted 0 old in 10.74 seconds
...
Complete!
```

Congratulations!

You now have a base installation in the image file you've created. The next steps are to configure the installation to operate inside Amazon EC2, and to customize the installation for your use.

Configure the Installed Operating System

The base operating system has now successfully been installed. You must now configure the networking and hard drives to work in the Amazon EC2 environment.

Configure the Network Interface

The Amazon EC2 environment provides a networking interface card that needs to be configured to provide external network access for the running instance. Edit (or create) the following file `/mnt/ec2-fs/etc/sysconfig/network-scripts/ifcfg-eth0`, making sure it contains at least the following information.

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

Note
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. In addition, this local hostname will be the same for all instances of the AMI, which may prove confusing.

Enable Networking

After configuring the network interface, you need to ensure that networking will come up when the system is started. To do this, ensure that (at least) the following appears in `/mnt/ec2-fs/etc/sysconfig/network`.

```
NETWORKING=yes
```

Set up Hard Drives in `/etc/fstab`

Amazon EC2 provides the instance with additional local storage by way of a disk drive on `/dev/sda2`. In addition, swap space is provided on `/dev/sda3`. To ensure both these are mounted at system start up time, add the following lines to `/mnt/ec2-fs/etc/fstab`:

```
/dev/sda2  /mnt      ext3    defaults        1 2
/dev/sda3  swap      swap    defaults        0 0
```

Configure Additional Services

Finally, make sure that all of your required services will be started at system start up time by allocating them to the appropriate system run levels. To enable the service `my-service` on multi-user and networked run levels, for example, execute:

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

Unmount the Loopback File

Your new installation has now been successfully installed and configured to operate in the Amazon EC2 environment. You may now unmount the image:

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

Bundling an AMI

A root file system image needs to be bundled as an AMI in order to be used with the Amazon EC2 service. The bundling process first compresses the image to minimize bandwidth usage and storage requirements. The compressed image is then encrypted and signed to ensure confidentiality of the data, and authentication against the creator. The encrypted image is finally split into manageable parts for upload. A manifest file is created containing a list of the image parts with their checksums. This chapter provides an overview of the AMI tools that automate this process and some examples of their use.

The AMI tools are three command-line utilities:

1. **ec2-bundle-image** bundles an existing AMI
2. **ec2-bundle-vol** creates an AMI from an existing machine or installed volume
3. **ec2-upload-bundle** uploads a bundled AMI to S3 storage

Installing the AMI Tools

The AMI tools are packaged as an RPM suitable for running on Fedora Core 3/4 with Ruby 1.8.2 (or greater) installed. On Fedora Core 4 Ruby can be installed by following the steps below. You will need root privileges to install the software. You can find the AMI tools RPM at .

First install Ruby using the yum package manager.

```
# yum install ruby
```

Install the `ec2-common-ruby` RPM which contains library functions that the AMI tools depend on. The `x`'s represent the major version, minor version and build number of the RPMs.

```
# rpm -i ec2-common-ruby-x.x-xxxx.i386.rpm
```

Install the AMI tools RPM.

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

Installation Issues

The AMI tools libraries install under `/usr/lib/site_ruby`. Ruby should pick up this path automatically, but if you see a load error when running one of the AMI utilities, it may be because Ruby isn't looking there. To fix this, add `/usr/lib/site_ruby` to Ruby's library path, which is set in the `RUBYLIB` environment variable.

Documentation

The manual describing the operation of each utility can be displayed by invoking it with the *`--manual`* parameter. For example:

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

Invoking a utility with the *`--help`* parameter displays a summary and list of command line parameters. For example:

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

Using the AMI Tools

Once a machine image has been created it must be bundled as an AMI for use with Amazon EC2, as follows. Use **ec2-bundle-image** to bundle an image that you have prepared in a loopback file, as described in the previous section.

```
# ec2-bundle-image -i my-image.img -k my-private-key.key -c my-x509-cert.cert
```

This will create the bundle files:

```
image.part.00
image.part.01
...
image.part.NN
image.manifest
```

Alternatively an AMI could be created by snapshotting the local machine root file system and bundling it all at once by using **ec2-bundle-vol**. (note: you will need to have root privileges to do this and SELinux must be disabled). Use **ec2-bundle-vol** to re-bundle a (modified) running instance of an existing AMI, as described in the previous section.

```
# ec2-bundle-vol -k my-private-key.key -s 1000 -u 495219933132
```

As with **ec2-bundle-image**, **ec2-bundle-vol** will create image parts files and a manifest file.

Note

If selinux is enabled when **ec2-bundle-vol** is run, the filesystem creation step may fail. Selinux should be disabled while this is done.

Uploading a Bundled AMI

The bundled AMI needs to be uploaded for storage in Amazon S3 before it can be accessed by Amazon EC2. Use **ec2-upload-bundle** to upload the bundled AMI that you created as described above. S3 stores data objects in buckets, which are similar in concept to directories. Buckets must have globally unique names. The **ec2-upload-bundle** utility will upload the bundled AMI to a specified bucket. If the specified bucket does not exist it will be created. However, if the specified bucket already exists, and belongs to another user, then **ec2-upload-bundle** will fail.

```
# ec2-upload-bundle -b my-bucket -m image.manifest -a my-aws-access-key-id -s
my-secret-key-id
```

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

Securing the Network

The Amazon EC2 service provides the ability to dynamically add and removed 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 compute resources to user-defined groups and define firewall rules for and in terms of these groups. As compute resources are added to or removed from groups, the appropriate rules are enforced. Similarly, if a group's rules are changed these changes are automatically applied to all members of the affected 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 may be useful if you have specific requirements not catered for by the distributed firewall.

Anticipated API changes

At present, the API calls for authorizing and revoking permissions are still under development. The remainder of this section outlines what you can depend on from this part of our API. The command line API tools expose only the subset of the functionality that is expected to remain unchanged.

Callers may depend on, now and in future, being able to grant permissions to

- source address ranges (specified with CIDRs, specific protocol and ports (or ICMP type/code)).
- source {user,group} tuples. No additional granularity, such as protocol and port (or ICMP type/code), should be expected.

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.

A group's rules may be modified at any time. The new rules are automatically enforced for all running, as well as for subsequently launched, instances affected by the change in rules.

Note: Currently there is a limit of one hundred rules per group.

Group Membership

When an AMI instance is launched it may be assigned membership to any number of groups.

If no groups are specified, the instance is assigned to the "default" group. This group can be modified, by you, like any other group you have created. Be default, this group allows all network traffic from other members of the "default" group and discards traffic from other IP addresses and groups.

Group Access Rights

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

Examples

We illustrate the use of the Amazon EC2 firewall in the following two examples. Note that we use the command line tools throughout the examples. The same results can be achieved using the SOAP API.

Default Group

1. Albert launches a copy of his favourite public AMI

   ```
   $ ec2-run-instances ami-eca54085
   RESERVATION     01927768      598916040194
   INSTANCE        cfd732a6      ami-eca54085            pending
   ```

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    USER
   598916040194    GRPNAME default
   ```

   and 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.com
   Starting nmap 3.81 ( http://www.insecure.org/nmap/ ) at 2006-08-07 15:42
   SAST
   All 100 scanned ports on domU-12-31-33-00-01-56.usma1.compute.amazonaws.com
   (216.182.228.116) are: filtered

   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      FROM
   CIDR    192.168.1.130/32
   ```

5. Repeating the port scan

   ```
   $ nmap -P0 -p1-100 domU-12-31-33-00-01-56.usma1.compute.amazonaws.com
   Starting nmap 3.81 ( http://www.insecure.org/nmap/ ) at 2006-08-07 15:43
   SAST
   Interesting ports on domU-12-31-33-00-01-56.usma1.compute.amazonaws.com
   (216.182.228.116):
   (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 that nasty people might gain access to 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 she 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"
   GROUP   apache  Mary's Apache group

   $ ec2-describe-group apache
   GROUP   598916040194    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      FROM
   CIDR    0.0.0.0/0

   $ ec2-describe-group apache
   GROUP   598916040194    apache  Mary's Apache group
   PERMISSION      598916040194    apache  ALLOWS  tcp     80      80
   FROM    CIDR    0.0.0.0/0
   ```

   She then launches seven instances of her web server AMI as members of this group

   ```
   $ ec2run ami-fba54092 -n 7 -g apache
   RESERVATION     01927768      598916040194
   INSTANCE        cfd732a6      ami-fba54092            pending
   ...

   $ ec2din cfd732a6
   RESERVATION     0592776c      598916040194
   INSTANCE        cfd732a6      ami-fba54092       domU-
   12-31-33-00-04-16.usma1.compute.amazonaws.com      running
   ...
   ```

   Having studied at the same school of paranoia as Albert, Mary does a port scan to confirm the permissions she just configured

   ```
   $ nmap -P0 -p1-100 domU-12-31-33-00-04-16.usma1.compute.amazonaws.com
   Starting nmap 3.81 ( http://www.insecure.org/nmap/ ) at 2006-08-07 16:21
   SAST
   Interesting ports on domU-12-31-33-00-04-16.usma1.compute.amazonaws.com
   (216.182.231.20):
   (The 99 ports scanned but not shown below are in state: filtered)
   PORT   STATE SERVICE
   80/tcp open  http

   Nmap finished: 1 IP address (1 host up) scanned in 33.409 seconds
   ```

   And then she tests to make sure her web server is contactable

   ```
   $ telnet domU-12-31-33-00-04-16.usma1.compute.amazonaws.com 80
   Trying 216.182.231.20...
   Connected to domU-12-31-33-00-04-16.usma1.compute.amazonaws.com
   (216.182.231.20).
   Escape character is '^]'.
   ```

   Excellent!
2. She now creates a separate group for her application server

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

then starts twenty instances as members of this group

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

and grants network access between her web server group and the application server group

```
$ ec2-authorize appserver -o apache -u 598916040194
GROUP           appserver
PERMISSION      appserver  ALLOWS  all                    FROM    USER
598916040194    GRPNAME apache
```

She checks to ensure access to her app server is indeed restricted by port scanning one of the app servers

```
$ nmap -P0 -p1-100 domU-12-31-33-00-03-D1.usma1.compute.amazonaws.com
Starting nmap 3.81 ( http://www.insecure.org/nmap/ ) at 2006-08-07 15:42
SAST
All 100 scanned ports on domU-12-31-33-00-03-D1.usma1.compute.amazonaws.com
(216.182.228.12) are: filtered

Nmap finished: 1 IP address (1 host up) scanned in 31.008 seconds
```

3. To confirm that her web servers have access to her application servers she needs to do a little extra work...
   a. She (temporarily) grants SSH access from her workstation to the web server group

      ```
      $ 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.usma1.compute.amazonaws.com 8080
      Trying 216.182.228.12...
      Connected to domU-12-31-33-00-03-D1 .usma1.compute.amazonaws.com
      (216.182.228.12).
      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
      ```

Creating the group for database servers and granting access to them from the application server group is left as an exercise for the reader ;-)

Tools and APIs

Below we highlight the most relevant command-line tools and SOAP API calls used to manipulate security groups. Please refer to the appropriate sections of this guide for the specific details.

Purpose	Command-line tool	SOAP API
List the rules belonging to specified groups	**`ec2-describe-group`**	DescribeSecurityGroups
Create a new security group	**`ec2-add-group`**	CreateSecurityGroup
Delete an existing security group	**`ec2-delete-group`**	DeleteSecurityGroup
Add an access rule to an existing security group	**`ec2-authorize`**	AuthorizeSecurityGroupIngress
Remove an access rule from an existing security group	**`ec2-revoke`**	RevokeSecurityGroupIngress

Using Instances

The instance is your basic computation building block. It is a medium-sized host that provides you with the same predictable performance you would expect from a physical host. You can run on as many or as few as you need at any given time. Each instance predictably provides the equivalent of a system with a 1.7Ghz Xeon CPU, 1.75GB of RAM, 160GB of local disk, and 250Mb/s of network bandwidth.

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.

Best Practices

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

- Do not rely on an instance's local storage for valuable, long-term data . Instances can fail, and when they fail, the data on the local disk is lost. You should 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 your instances perform. For "internet applications" you may choose to define one image for database instances and one image for your webservers. Image creation and storage are cheap and easy operations. Individualize and customize as necessary. Keeping your images specialized will mean that the resulting AMIs can be smaller. Smaller AMIs will boot considerably faster.
- Monitor the health of your instances. Make your instances work for you by monitoring each other. You may choose to create an image which contains one of the various open-source monitoring tools such as Nagios or OpenNMS. Each worker instance, based on your other images, might then report its health to your monitoring instance.
- Keep your Amazon EC2 firewall permissions as restrictive as possible. Only open up permissions you need to open. Use separate groups to deal with instances that have different network ingress requirements. Consider using additional security measures inside your instance including your own firewall. If you need to login interactively (ssh), consider creating a bastion security group that allows external login, while the remainder of your instances are in a group that does not allow external login.

Command Line Tool 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. Optional parameters are indicated by means of flags. Flags typically come in a short and long form, although not all flags exist in both forms. In their short form, flags are a single character prefixed with a single dash. In their long form, flags use a longer, more expressive name prefixed with a double dash. Optional parameters typically have default values, or may be required only when other optional parameters are specified, and order is unimportant. For all remaining parameters order does matter.

A number of command line options apply to all of the command line tools. These are covered below and, for reasons of brevity, are not included in the description of each of the specific tools.

Errors

Any service errors encountered by the command line tools will be passed straight through from the API. A list of these errors can be seen in the section called “API Error Codes”.

Common Options

All command line tools covered in the following sections accept a common set of optional parameters as follows:

Element Name	Definition	Valid Values/ Types	Example
-U URL	URL is the uniform resource locator of the Amazon EC2 web service entry point. This option defaults to the value of the `EC2_URL` environment variable, or http://ec2.amazonaws.com if that is not set.	URL	**`-U http://ec2.amazonaws.com`**
-K EC2-PRIVATE-KEY	The private key to use when constructing requests to Amazon EC2. This parameter defaults to the value of the `EC2_PRIVATE_KEY` environment variable.	File name	**`-K pk-HKZYK-TAIG2ECMXYIBH3HXV4ZBZQ55CLO.pem`**
-C EC2-CERT	The X509 certificate to use when constructing requests to Amazon EC2. This parameter defaults to the value of the `EC2_CERT` environment variable.	File name	**`-C cert-HKZYK-TAIG2ECMXYIBH3HXV4ZBZQ55CLO.pem`**
-v	Increase output verbosity. This will print the SOAP request and response on the command line. This is particularly useful if you're trying to build your own tools to talk directly to our SOAP API.	N/A	N/A
--debug	Print internal debugging information. This is intended to assist us to troubleshoot problems.	N/A	N/A
-?	Show help.	N/A	N/A
-	If **-** is specified as an argument to one of the parameters, a list of arguments will be read from stdin. This is useful for piping the output of one command into the input of another.	N/A	**`ec2-describe-instances | grep running | cut -f 2 | ec2-terminate-instances -i -`**

ec2-register

SYNOPSIS

```
ec2-register MANIFEST
```

DESCRIPTION

Registers the Amazon Machine Image (AMI) described by the named MANIFEST file, generating a new Amazon Machine Image (AMI) ID. MANIFEST must specify a location of a manifest file in Amazon S3 and must be of the form `bucket/object`.

OUTPUT

The image ID that was assigned by Amazon EC2 is displayed.

Errors are displayed on stderr.

EXAMPLE

```
$ ec2-register mybucket/image.manifest
IMAGE ami-78a54011
```

SEE ALSO

- RegisterImage
- ec2-deregister
- ec2-describe-images

ec2-describe-images

SYNOPSIS

`ec2-describe-images [AMI ...]`

DESCRIPTION

Describes the current state of each AMI specified on the command line. If no AMIs are explicitly listed then all AMIs owned by the current user, and all public AMIs are included in the output.

OUTPUT

A table containing the following information is returned:

- A record type identifier ("IMAGE")
- image identifier
- manifest location
- user identifier of the user that registered the image
- image status
- `public` or `private` indicating whether or not the image is visible to all users

Errors are displayed on stderr.

EXAMPLE

```
$ ec2-describe-images ami-78a54011
IMAGE ami-78a54011 powerdns/image.manifest 495219933132 available private
```

SEE ALSO

- DescribeImages
- ec2-register
- ec2-deregister

ec2-deregister

SYNOPSIS

```
ec2-deregister AMI
```

DESCRIPTION

The AMI identified is deregistered. This AMI may no longer be used to launch new instances. The AMI is not deleted from Amazon S3

OUTPUT

A table containing the following information is returned:

- A record type identifier ("IMAGE")
- the image identifier that was deregistered

Errors are displayed on stderr.

EXAMPLE

```
$ ec2-deregister ami-4fa54026
IMAGE ami-4fa54026
```

SEE ALSO

- DeregisterImage
- ec2-register
- ec2-describe-images

ec2-add-keypair

SYNOPSIS

ec2-add-keypair KEY

DESCRIPTION

A new 2048 bit RSA key pair is created 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 an error is returned.

OUTPUT

A table containing the following information is returned:

- Output type identifier ("KEYPAIR").
- Keypair name.
- Private key fingerprint.
- Private key. This value is displayed on a new line.

Errors are displayed on stderr.

EXAMPLE

```
$ 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-----
MIICXAIBAAKBgQC2i/Sgs5BGGd4sunpYQfEkcprgzP9M/hnVJTc1j0nZBeIE2JBuLRSNoqkO7Gw8
nBcdNptaLedzqN8t78jGkX1TPWVAKJTfxRSvU/oViGJaRqIBar0Mpc/wC27kyzHezUNS5+mvONb3
4h/j2EZwDLY75Uxrpka0aN6OkvyIP5gYMQIDAQABAoGAOKH65tBOdjEYSHAh/LeYhGI5wnxWyCAd
C49cLXWix32XvUEircu2kKpiIIsgmT0jvqBuWe/b2noNo0a81z3TzzRYyLvn5J8mUlL6a8nsssQ3
xCHkGM+SE7ZzfBS5WUkbh5Exd3ZXKfCJvJW6auOzJ581JB5yUNbqixWzHuQGGAECQQDwq4LQoyb5
OVSpZwSy+GW/p0yRsqRp89ECNQ+hySGBjkSXBcbt75C+5ebo88/V2V4QOGGa0T0tMsMgKTJ8oukh
AkEAwiyoFM0Zwk0Os3rBZ8PyZoNW5e5SBwrEbLRv4JCaNiQme0ighsDr2bL/nGLI7p13g22+9REM
i/WAmsln50H9EQJASMun6tGepT2pFQBbFIM7y4egCmXdg0rDSoagLtB2eQh+SKvvquKOhp9lg8rT
b5yq7f8PztNBTN2Q1baAVeC04QJAGgN5kS/ZH5rLOWhcuNYbh3hZD/zZqG/c2ONjiaZVwqMdNK8K
MoNuFYBRllX1rWITPNxbFOHv2GBPlm0dKnJAwQJBAOgwjgLY3UpXFX9ZvG4RGEYgfui49Ffz10CH
5sSZpsFYn42E6a2NUJeL4hTzfbGTQ8iCIVjOXFH/9XLTDCNQEPM=
-----END RSA PRIVATE KEY-----
```

SEE ALSO

- CreateKeypair
- ec2-describe-keypairs
- ec2-delete-keypair

ec2-describe-keypairs

SYNOPSIS

```
ec2-describe-keypairs [KEY ...]
```

DESCRIPTION

Describes the current state of each KEY specified on the command line. If no KEYs are explicitly listed then all KEYs owned by the current user are included in the output.

OUTPUT

A table containing the following information is returned:

- A output type identifier ("KEYPAIR")
- Keypair identifier
- Private key fingerprint

Errors are displayed on stderr.

EXAMPLE

```
$ 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
```

SEE ALSO

- DescribeKeypairs
- ec2-add-keypair
- ec2-delete-keypair

ec2-delete-keypair

SYNOPSIS

```
ec2-delete-keypair KEY
```

DESCRIPTION

Deletes the named KEY, purging the public key from Amazon EC2

OUTPUT

A table containing the following information is returned:

- Output type identifier ("KEYPAIR").
- Identifier of the deleted keypair.
- Private key fingerprint.

Errors are displayed on stderr.

EXAMPLE

```
$ ec2-delete-keypair gsg-keypair
KEYPAIR gsg-keypair
```

SEE ALSO

- DeleteKeypair
- ec2-add-keypair
- ec2-describe-keypairs

ec2-fingerprint-key

SYNOPSIS

```
ec2-fingerprint-key KEYFILE
```

DESCRIPTION

Computes and displays the fingerprint for a private key produced by Amazon EC2. KEYFILE must be the path to a file containing an unencrypted PEM encoded PKCS#8 private key.

This operation is performed entirely on the client-side. Network access is not required.

OUTPUT

A key fingerprint. This is formatted as a hash digest with each octet separated by a colon.

Errors are displayed on stderr.

EXAMPLE

```
$ 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
```

SEE ALSO

- ec2-create-keypair
- ec2-describe-keypairs

ec2-run-instances

SYNOPSIS

```
ec2-run-instances AMI [-n INSTANCE_COUNT] [-g GROUP [-g GROUP ...]] [-k KEY]
```

DESCRIPTION

Launches one or more instances of the specified AMI. An optional security group in which new instances should be launched may be specified. If no group is specified instances are launched in the `default` group. A keypair name may also be specified. If provided, the public key associated with this keypair will be made available to the instances at boot time.

OUTPUT

A table containing the following information is returned:

- Output type identifier ("INSTANCE")
- Instance ID which uniquely identifies each running instance.
- AMI ID of the image the instance is based on.
- DNS name associated with the instance (only present for instances in the `running` state).
- Instance state. This will in most cases be *`pending`* which indicates that the instance is being prepared for launch.
- Key name. If a key was associated with the instance at launch it's name will be displayed in this column.

Errors are displayed on stderr.

OPTIONS

Option	Definition	Required?	Example
`-n INSTANCE_COUNT`	The number of instances to launch. If not specified, a value of `1` will be assumed. If it is not possible to launch at least this many instances (due to a lack of capacity or funds), no instances will be launched. If specified as a range (min-max) Amazon EC2 will try to launch as many instances as possible, up to max, but will launch no fewer than min instances.	No	-n 5
`-g GROUP`	The security group(s) within which the instance(s) should be run. This determines the ingress firewall rules that will be applied to the instances. By default instances will run in the user's default group. If more than one group is specified, the security policy of the instances will be the union of the security policies of the specified groups.	No	-g fooGroup

Option	Definition	Required?	Example
-k KEY	The keypair to make available to these instances at boot time.	No	-k fooKeyPair

EXAMPLE

```
$ ec2-run-instances ami-6ba54002 -n 5
INSTANCE i-3ea74257 ami-6ba54002 pending
INSTANCE i-31a74258 ami-6ba54002 pending
INSTANCE i-31a74259 ami-6ba54002 pending
INSTANCE i-31a7425a ami-6ba54002 pending
INSTANCE i-31a7425b ami-6ba54002 pending
INSTANCE i-31a7425c ami-6ba54002 pending
```

SEE ALSO

- RunInstances
- ec2-terminate-instances
- ec2-describe-instances
- ec2-add-keypair

ec2-describe-instances

SYNOPSIS

```
ec2-describe-instances [INSTANCEID ...]
```

DESCRIPTION

Describes the current state of each instance indicated by the respective INTANCEID specified on the command line. If no instances are explicitly listed then all instances owned by the current user are included in the output.

OUTPUT

A table containing the following information is returned:

- Output type identifier ("RESERVATION", "INSTANCE")
- Instance ID which uniquely identifies each running instance.
- AMI ID of the image the instance is based on.
- DNS name associated with the instance (only present for instances in the `running` state).
- Instance state.
- Key name. If a key was associated with the instance at launch it's name will be displayed in this column.

Errors are displayed on stderr.

EXAMPLE

```
$ ec2-describe-instances
RESERVATION r-15a4417c 495219933132
INSTANCE i-3ea74257 ami-6ba54002 domU-
12-31-33-00-00-01.dc3.compute.amazonaws.com running
INSTANCE i-31a74258 ami-6ba54002 domU-
12-31-33-00-00-02.dc3.compute.amazonaws.com running
```

SEE ALSO

- DescribeInstances
- ec2-run-instances
- ec2-terminate-instances

ec2-terminate-instances

SYNOPSIS

```
ec2-terminate-instances INSTANCEID [INSTANCEID ...]
```

DESCRIPTION

All instances indicated by the respective INTANCEID specified on the command line are terminated. At least one INSTANCEID must be specified.

OUTPUT

A table containing the following information is returned:

- 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.

Errors are displayed on stderr.

EXAMPLE

```
$ ec2-terminate-instances i-3ea74257
INSTANCE i-3ea74257 running shutting-down
```

SEE ALSO

- TerminateInstances
- ec2-run-instances
- ec2-describe-instances

ec2-add-group

SYNOPSIS

```
ec2-add-group GROUP -d DESCRIPTION
```

DESCRIPTION

Creates a new security group named GROUP. Group names must be unique per user.

OUTPUT

A table containing the following information is returned:

- Output type identifier ("GROUP").
- Group name.
- Group description.

Errors are displayed on stderr.

OPTIONS

Option	Definition	Required?	Example
-d DESCRIPTION	Description of the group. This is informational only.	Yes	-d 'Web servers'

EXAMPLE

```
$ ec2-add-group websrv -d 'Web servers'
GROUP websrv Web servers
```

SEE ALSO

- CreateSecurityGroup
- ec2-describe-groups
- ec2-delete-group
- ec2-authorize
- ec2-revoke

ec2-describe-groups

SYNOPSIS

```
ec2-describe-groups [GROUP ...]
```

DESCRIPTION

Describes the current state of each GROUP specified on the command line. If no GROUPs are explicitly listed then all GROUPs owned by the current user are included in the output.

OUTPUT

A table containing the following information is returned:

- Output type identifier ("GROUP", "PERMISSION").
- User ID of group owner.
- Group name.
- Description of the group.
- Firewall rule.

Errors are displayed on stderr.

EXAMPLE

```
$ ec2-describe-groups websrv
GROUP 495219933132 websrv Web servers
PERMISSION 495219933132 websrv ALLOWS tcp 80 80 FROM CIDR 0.0.0.0/0
```

SEE ALSO

- DescribeSecurityGroups
- ec2-add-group
- ec2-delete-group
- ec2-authorize
- ec2-revoke

ec2-delete-group

SYNOPSIS

```
ec2-delete-group GROUP
```

DESCRIPTION

Deletes the named GROUP.

OUTPUT

A table containing the following information is returned:

- Output type identifier ("GROUP").
- Name of the deleted group.

Errors are displayed on stderr.

EXAMPLE

```
$ ec2-delete-group websrv
GROUP websrv
```

SEE ALSO

- DeleteSecurityGroup
- ec2-add-group
- ec2-describe-groups
- ec2-authorize
- ec2-revoke

ec2-authorize

SYNOPSIS

```
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 security group named GROUP. If no source host, group or subnet is provided, requests from any source address will be honored.

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.

Errors are displayed on stderr.

OPTIONS

Option	Definition	Required?	Example
`-P PROTOCOL`	The protocol to allow. This can be `tcp`, `udp` or `icmp`. This option only applies when specifying a CIDR subnet as the source.	Yes	-P tcp
`-p PORT_RANGE`	For the TCP or UDP protocols, this specifies the range of ports to allow. This may be specified as a single integer or as a range (min-max). This option only applies when specifying a CIDR subnet as the source.	Yes	-p 80
`-t ICMP_TYPE_CODE`	For the ICMP protocol, the ICMP type and code must be specified. This must be specified as type:code where both are integers. Type or code (or both) may be specified as -1 which is a wildcard. This option only applies when specifying a CIDR subnet as the	Yes	-t 2:5

Option	Definition	Required?	Example
	source.		
-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.	No	-u 495219933132
-o SOURCE_GROUP	The network source from which traffic is to be authorized specified as a security Group. See the description of the *-u* parameter for group owner information.	No	-o headoffice
-s SOURCE_SUBNET	The network source from which traffic is to be authorized specified as a CIDR Subnet range.	No	-s 205.192.8.45/24

EXAMPLE

```
$ 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
```

SEE ALSO

- AuthorizeSecurityGroupIngress
- ec2-add-group
- ec2-describe-groups
- ec2-delete-group
- ec2-revoke

ec2-revoke

SYNOPSIS

```
ec2-revoke GROUP [-P PROTOCOL] (-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 create that rule.

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.

Errors are displayed on stderr.

OPTIONS

Option	Definition	Required?	Example
`-P` *`PROTOCOL`*	The protocol to allow. This can be`tcp`, `udp` or `icmp`. This option only applies when specifying a CIDR subnet as the source.	Yes	-P tcp
`-p` *`PORT_RANGE`*	The range of ports to revoke. This may be specified as a single integer or as a range (min-max). This option only applies when specifying a CIDR subnet as the source.	Yes	-p 80
`-t` *`ICMP_TYPE_CODE`*	If the protocol is ICMP, the ICMP type and code must be specified. This must be specified as type:code where both are integers. Type or code (or both) may be specified as -1 which acts as a wildcard. This option only applies when specifying a CIDR subnet as the source.	Yes	-t 2:5

Option	Definition	Required?	Example
-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.	No	-u 495219933132
-o SOURCE_GROUP	The network source from which traffic is to be revoked specified as a security Group. See the description of the *-u* parameter for group owner information.	No	-o outsideworld
-s SOURCE_SUBNET	The network source from which traffic is to be revoked specified as a CIDR Subnet range.	No	-s 205.192.8.45/24

EXAMPLE

```
$ 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
```

SEE ALSO

- RevokeSecurityGroupIngress
- ec2-add-group
- ec2-describe-groups
- ec2-delete-group
- ec2-authorize

Using the SOAP API

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 datatypes that may appear in SOAP requests and responses. For more information on WSDL and SOAP, please see the references in the section called “Additional Web Services References”.

All schemas have a version number. The version number appears in the URL of a schema file, and in a schema's target namespace. The latest version is 2006-06-26. Upgrading is made 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. Once customer transition to the new version is complete, the older versions will be retired.

The Amazon EC2 services API WSDL can be found at URLs of the form 'http://ec2.amazonaws.com/doc/VERSION/ec2.wsdl' where VERSION indicates the version of the API. The current API version is 2006-06-26 and can thus be found at URL http://ec2.amazonaws.com/doc/2006-06-26/AmazonEC2.wsdl

Making Requests

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 BinarySecurityToken profile, consisting of an X.509 certificate with an RSA public key.

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 may be used. Languages known to have such support include Perl, Python, Java and C++. Currently we only supply java libraries for our API but expect to release additional language bindings in the future.

Request Authentication

The following is an insecure request to run instances:

```
<RunInstances xmlns="http://ec2.amazonaws.com/doc/2006-06-26">
    <instancesSet>
        <item>
            <imageId>ami-60a54009</imageId>
            <minCount>1</minCount>
            <maxCount>3</maxCount>
        </item>
    </instancesSet>
    <groupSet/>
</RunInstances>
```

In order to secure the request, we must add the BinarySecurityToken element mentioned above. 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:

```
<SOAP-ENV:Envelope xm-
lns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <wsse:Security xm-
lns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd">
      <wsse:BinarySecurityToken
      xm-
lns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-u
tility-1.0.xsd"
      Encoding-
Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-se
curity-1.0#Base64Binary"
ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token
-profile-1.0#X509v3"
      wsu:Id="CertId-1064304">....many, many lines of base64 encoded
      X.509 certificate...</wsse:BinarySecurityToken>
      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:SignedInfo>
          <ds:CanonicalizationMethod Al-
gorithm="http://www.w3.org/2001/10/xml-exc-c14n#"></ds:CanonicalizationMethod
>
          <ds:SignatureMethod Al-
gorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"></ds:SignatureMethod>
          <ds:Reference URI="#id-17984263">
            <ds:Transforms>
              <ds:Transform Al-
gorithm="http://www.w3.org/2001/10/xml-exc-c14n#"></ds:Transform>
            </ds:Transforms>
            <ds:DigestMethod Al-
gorithm="http://www.w3.org/2000/09/xmldsig#sha1"></ds:DigestMethod>
            <ds:DigestValue>0pjZ1+TvgPf6uG7o+Yp3l2YdGZ4=</ds:DigestValue>
          </ds:Reference>
          <ds:Reference URI="#id-15778003">
            <ds:Transforms>
              <ds:Transform Al-
gorithm="http://www.w3.org/2001/10/xml-exc-c14n#"></ds:Transform>
            </ds:Transforms>
            <ds:DigestMethod Al-
gorithm="http://www.w3.org/2000/09/xmldsig#sha1"></ds:DigestMethod>
            <ds:DigestValue>HhRbxBBmc2OO348f8nLNZyo4AOM=</ds:DigestValue>
          </ds:Reference>
        </ds:SignedInfo>
<ds:SignatureValue>bmVx24Qom4kd9QQtclxWIlgLk4QsQBPaKESi79x479xgbO9PEStXMiHZuB
```

```
Ai9luuKdNTcfQ8UE/d
        jjHKZKEQR-
COlLVy0Dn5ZL1RlMHsv+OzJzzvIJFTq3LQKNrzJzsNe</ds:SignatureValue>
        <ds:KeyInfo Id="KeyId-17007273">
          <wsse:SecurityTokenReference
              xm-
lns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-u
tility-1.0.xsd" wsu:Id="STRId-22438818">
            <wsse:Reference URI="#CertId-1064304"
ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token
-profile-1.0#X509v3">
            </wsse:Reference>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
      <wsu:Timestamp
          xm-
lns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-u
tility-1.0.xsd" wsu:Id="id-17984263">
        <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>
```

Let's take a quick look at the most important elements in case you are matching this against requests generated by Amazon EC2 supplied libraries, or those of another vendor.

- BinarySecurityToken - contains the X.509 certificate in base64 encoded PEM format.
- Signature - contains XML digital signature created using the canonicalization, signature algorithm, and digest method described within.
- Timestamp - Any request is only valid to Amazon EC2 within 5 minutes of this value. Used to prevent replay attacks.

Understanding Responses

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 datatypes with be named according to the operation performed and whether the datatype is a container (may 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/2006-06-26">
  <reservationId>r-47a5402e</reservationId>
  <ownerId>UYY3TLBUXIEON5NQVUUX6OMPWBZIQNFM</ownerId>
  <groupSet>
    <item>
      <groupId>default</groupId>
    </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>
    </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>
    </item>
  </instancesSet>
</RunInstancesResponse>
```

Additional Web Services References

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

API Reference

The Amazon EC2 API allows developers to launch and control instances from their own programs. This API follows the SOAP and WSDL standards, making it accessible from any language for which suitable libraries exist.

This section discusses the messages available in the Amazon EC2 SOAP API, the semantics of those calls and the parameters that must be supplied. Examples of SOAP requests and responses are also provided.

We recommend you familiarize yourself with the conventions we've used in describing the API.

API Conventions

Overview

This topic discusses the conventions used in the Amazon EC2 SOAP API reference. This includes terminology, notation and any abbreviations used to illuminate the API.

The API reference is broken down into a collection of *Operations* and *Data Types*.

Operations

Operations 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 authenticate the caller. 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 the following location: http://ec2.amazonaws.com/doc/2006-06-26/AmazonEC2.wsdl. 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[]`:

```
<imagesSet>
  <item>
    <imageId>ami-61a54008</imageId>
  </item>
  <item>
    <imageId>ami-61b54608</imageId>
  </item>
</imagesSet>
```

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

```
<instancesSet>
    <item>
        <imageId>ami-60a54009</imageId>
        <minCount>10</minCount>
        <maxCount>30</maxCount>
    </item>
    <item>
        <imageId>ami-60b54209</imageId>
        <minCount>5</minCount>
        <maxCount>20</maxCount>
    </item>
</instancesSet>
```

API Error Codes

Summary of Error Codes

SOAP Fault-Code/ Error-Code	Definition	Notes
`Client.AuthFailure`	User not authorized.	Common cause is trying to run an AMI for which you do not have permission.
`Client.InvalidManifest`	Specified AMI has an unparsable Manifest.	
`Client.InvalidAMIID`	Specified AMI ID is not valid.	If this error contains "Malformed" it further implies your id is not of the valid format.
`Client.InvalidInstanceID`	Specified instance ID is not valid.	If this error contains "Malformed" it further implies your id is not of the valid format.
`Client.InvalidGroupName`	Specified group name is invalid.	
`Client.InvalidReservationID`	Specified reservation ID is invalid.	If this error contains "Malformed" it further implies your id is not of the valid format.
`Client.InvalidGroup.Duplicate`	Attempt to create a duplicate group.	
`Client.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.
`Client.InvalidGroup.InUse`	Specific group can not be deleted because it is in use.	

SOAP Fault-Code/ Error-Code	Definition	Notes
`Client.InvalidParameterCombination`	RunInstances was called with minCount and maxCount set to 0 or minCount > maxCount.	
`Server.InternalError`	Internal Error.	Should not occur. Please let us know. Try to reproduce.
`Server.InsufficientInstanceCapacity`	Not enough available instances to satisfy your minimum request.	You can lower your request or wait for additional capacity to become available.
`Server.Unavailable`	Indicates the server is overloaded and cannot handle request.	

Operations

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

RegisterImage

The `RegisterImage` operation registers an AMI with Amazon EC2. Images must be registered before they can be launched.

Each AMI is associated with an unique ID which is provided by the EC2 service via the Registerimage operation. As part of the registration process, Amazon EC2 will retrieve the specified image manifest from Amazon S3 and verify that the image is owned by the user requesting image registration.

The image manifest is retrieved once and stored within the Amazon EC2 network. Any modifications to an image in Amazon S3 invalidate this registration. If you do have to make changes and upload a new image deregister the previous image and register the new image.

Request Parameters

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

Element Name	Definition	Required?	Type
`imageLocation`	Full path to your AMI manifest in Amazon S3 storage.	Yes	`xsd:string`

Response Tags

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

Element Name	Definition	Type
`imageId`	Unique ID of the newly registered machine image.	`xsd:string`

Sample Request

```
<RegisterImage xmlns="http://ec2.amazonaws.com/doc/2006-06-26">
  <imageLocation>/mybucket/myimage.manifest</imageLocation>
</RegisterImage>
```

Sample Response

```
<RegisterImageResponse xmlns="http://ec2.amazonaws.com/doc/2006-06-26">
  <imageId>ami-61a54008</imageId>
</RegisterImageResponse>
```

Related Operations

- DescribeImages
- DeregisterImage

DescribeImages

The `DescribeImages` operation returns information about AMIs available for use by the user making the request. This includes both public AMIs (those available for any user to launch) and private images (those available for launch by the image owner only).

An optional list of AMI IDs may be provided to request information for those AMIs only. If no AMI IDs are provided, information of all relevant AMIs will be returned. If an image is specified that does not exist a fault is returned. If an image is specified that exists but is not owned by the user making the request, then that image will not be included in the returned results.

Deregistered images will be included in the returned results for an unspecified interval subsequent to deregistration.

Request Parameters

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

Element Name	Definition	Required?	Type
`imageSet`	AMI IDs to describe.	Yes (but may be empty)	`xsd:string[]`

Response Tags

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

Element Name	Definition
`imagesSet`	A list of image descriptions

Sample Request

```
<DescribeImages xmlns="http://ec2.amazonaws.com/doc/2006-06-26">
  <imagesSet>
    <item>
      <imageId>ami-61a54008</imageId>
    </item>
  </imagesSet>
</DescribeImages></SOAP-ENV:Body></SOAP-ENV:Envelope>
```

Sample Response

```
<DescribeImagesResponse xmlns="http://ec2.amazonaws.com/doc/2006-06-26">
  <imagesSet>
    <item>
      <imageId>ami-61a54008</imageId>
      <imageLocation>aes-ttylinux/image.manifest</imageLocation>
      <imageState>available</imageState>
      <imageOwnerId>UYY3TLBUXIEON5NQVUUX6OMPWBZIQNFM</imageOwnerId>
      <isPublic>false</isPublic>
    </item>
  </imagesSet>
</DescribeImagesResponse>
```

Related Operations

- DescribeInstances

DeregisterImage

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

Request Parameters

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

Element Name	**Definition**	**Required?**	**Type**
`imageId`	Unique ID of a machine image, returned by a call to RegisterImage or DescribeImages.	Yes	`xsd:string`

Response Tags

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

Element Name	**Definition**
`return`	`true` if deregistration succeeded, otherwise `false`.

Sample Request

```
<DeregisterImage xmlns="http://ec2.amazonaws.com/doc/2006-06-26">
    <imageId>ami-61a54008</imageId>
</DeregisterImage>
```

Sample Response

```
<DeregisterImageResponse xmlns="http://ec2.amazonaws.com/doc/2006-06-26">
  <return>true</return>
```

```
</DeregisterImageResponse>
```

Related Operations

- RegisterImage
- DescribeImages

CreateKeyPair

The `CreateKeyPair` operation creates a new 2048 bit RSA keypair and returns a unique ID that can be used to reference this keypair when launching new instances.

Request Parameters

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

Element Name	Definition	Required?	Type
`keyName`	A unique name for this key.	Yes	`xsd:string`

Response Tags

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

Element Name	Definition
`keyName`	The key name provided in the original request.
`keyFingerprint`	A SHA-1 digest of the DER encoded private key.
`keyMaterial`	An unencrypted PEM encoded RSA private key.

Sample Request

```
<CreateKeyPair xmlns="http://ec2.amazonaws.com/doc/2006-06-26">
   <keyName>example-key-name</keyName>
</CreateKeyPair>
```

Sample Response

```
<CreateKeyPairResponse xmlns="http://ec2.amazonaws.com/doc/2006-06-26">
   <keyName>example-key-name</keyName>
<keyFingerprint>1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:f1:6f</
keyFingerprint>
   <keyMaterial>-----BEGIN RSA PRIVATE KEY-----
MIICXAIBAAKBgQC2i/Sgs5BGGd4sunpYQfEkcprgzP9M/hnVJTc1j0nZBeIE2JBuLRSNoqkO7Gw8
nBcdNptaLedzqN8t78jGkX1TPWVAKJTfxRSvU/oViGJaRqIBar0Mpc/wC27kyzHezUNS5+mvONb3
4h/j2EZwDLY75Uxrpka0aN6OkvyIP5gYMQIDAQABAoGAOKH65tBOdjEYSHAh/LeYhGI5wnxWyCAd
```

```
C49cLXWix32XvUEircu2kKpiIIsgmT0jvqBuWe/b2noNo0a81z3TzzRYyLvn5J8mUlL6a8nsssQ3
xCHkGM+SE7ZzfBS5WUkbh5Exd3ZXKfCJvJW6auOzJ581JB5yUNbqixWzHuQGGAECQQDwq4LQoyb5
OVSpZwSy+GW/p0yRsqRp89ECNQ+hySGBjkSXBcbt75C+5ebo88/V2V4QOGGa0T0tMsMgKTJ8oukh
AkEAwiyoFM0Zwk0Os3rBZ8PyZoNW5e5SBwrEbLRv4JCaNiQme0ighsDr2bL/nGLI7p13g22+9REM
i/WAmsln50H9EQJASMun6tGepT2pFQBbFIM7y4egCmXdg0rDSoagLtB2eQh+SKvvquKOhp9lg8rT
b5yq7f8PztNBTN2Q1baAVeC04QJAGgN5kS/ZH5rLOWhcuNYbh3hZD/zZqG/c2ONjiaZVwqMdNK8K
MoNuFYBRllX1rWITPNxbFOHv2GBPlm0dKnJAwQJBAOgwjgLY3UpXFX9ZvG4RGEYgfui49Ffz10CH
5sSZpsFYn42E6a2NUJeL4hTzfbGTQ8iCIVjOXFH/9XLTDCNQEPM=
-----END RSA PRIVATE KEY-----</keyMaterial>
</CreateKeyPairResponse>
```

Related Operations

- DescribeKeyPairs
- DeleteKeyPair
- RunInstances

DescribeKeyPairs

The `DescribeKeyPairs` operation returns information about keypairs available for use by the user making the request. Selected keypairs may be specified or the list may be left empty if information for all registered keypairs is required.

Request Parameters

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

Element Name	Definition	Required?	Type
keySet	Keypair IDs to describe.	Yes (but may be empty)	`xsd:string[]`

Response Tags

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

Element Name	Definition
keySet	A list of keypair descriptions

Sample Request

```
<DescribeKeyPairs xmlns="http://ec2.amazonaws.com/doc/2006-06-26">
  <keySet>
      <item>
         <keyName>example-key-name</keyName>
      </item>
   </keySet>
</DescribeKeyPairs></SOAP-ENV:Body></SOAP-ENV:Envelope>
```

Sample Response

```
<DescribeKeyPairsResponse xmlns="http://ec2.amazonaws.com/doc/2006-06-26">
  <keySet>
    <item>
      <keyName>example-key-name</keyName>
<keyFingerprint>1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:f1:6f</
keyFingerprint>
    </item>
  </keySet>
</DescribeKeyPairsResponse>
```

Related Operations

- CreateKeypair
- DeleteKeypair
- RunInstances

DeleteKeyPair

The `DeleteKeyPair` operation deletes a keypair.

Request Parameters

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

Element Name	Definition	Required?	Type
`keyName`	Name of the keypair to delete.	Yes	`xsd:string`

Response Tags

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

Element Name	Definition
`return`	`true` if the key was successfully deleted.

Sample Request

```
<DeleteKeyPair xmlns="http://ec2.amazonaws.com/doc/2006-06-26">
    <keyName>example-key-name</keyName>
</DeleteKeyPair>
```

Sample Response

```
<DescribeKeyPairs xmlns="http://ec2.amazonaws.com/doc/2006-06-26">
```

```
   <return>true</return>
</DescribeKeyPairs></SOAP-ENV:Body></SOAP-ENV:Envelope>
```

Related Operations

- CreateKeyPair
- DescribeKeyPairs

RunInstances

The `RunInstances` operation launches a specified number of instances.

A call to `RunInstances` is guaranteed to start no fewer than the requested minimum for each AMI specified. If you have insufficient credit, or if there is insufficient capacity available then no instances will be started. Amazon EC2 will make a best effort attempt to satisfy the requested maximum values. If you have sufficient credit to cover the specified minimum values but not the maximum values then credit available beyond that required to cover the specified minimums will be allocated to each image specified in a round robin fashion.

As an example, consider a request to launch two images (A and B), with minimum and maximum values of (5,10) and (20, 40) respectively.

If you have credit available for less than 25 instances then no instances will be launched (since the minimums of 5 and 20 cannot both be satisfied). Similarly, if there is insufficient capacity for 25 instances then no instances will be launched.

If you have credit available (or if there is capacity available) for only 30 instances then 5 instances of A and 20 instances of B will be launched. The remaining 5 instances worth of credit (or capacity) will be allocated in round robin fashion.

Every instance is launched in a security group. This may be specified as part of the launch request. If a security group is not indicated then instances are started in a the default security group.

An optional keypair ID may be provided for each image in the launch request. All instances that are created from images for which this is provided will have access to the associated public key at boot time (detailed below). This key may be used to provide secure access to an instance of an image on a per-instance basis. Amazon EC2 public images make use of this functionality to provide secure passwordless access to instances (and launching those images without a keypair ID will leave them inaccessible).

The public key material is made available to the instance at boot time by placing it in a file named `openssh_id.pub` 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 time (as part of `rclocal`, for example) allowing for secure password-less access. As the need arises, other formats will also be considered.

Request Parameters

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

Element Name	Definition	Required?	Type
`instancesSet`	Description of the instances to launch.	Yes	`ec2:RunInstanceItem`

Element Name	Definition	Required?	Type
			Type[]
groupSet	Description of the security groups to associate the instances with.	Yes	ec2:GroupSetType[]

Response Tags

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

Element Name	Definition
RunInstancesResponse	Status information about the instances started.

Sample Request

```
<RunInstances xmlns="http://ec2.amazonaws.com/doc/2006-06-26">
    <instancesSet>
        <item>
            <imageId>ami-60a54009</imageId>
            <minCount>1</minCount>
            <maxCount>3</maxCount>
            <keyName>example-key-name</keyName>
        </item>
    </instancesSet>
    <groupSet/>
</RunInstances>
```

Sample Response

```
<RunInstancesResponse xmlns="http://ec2.amazonaws.com/doc/2006-06-26">
  <reservationId>r-47a5402e</reservationId>
  <ownerId>UYY3TLBUXIEON5NQVUUX6OMPWBZIQNFM</ownerId>
  <groupSet>
    <item>
      <groupId>default</groupId>
    </item>
  </groupSet>
  <instancesSet>
    <item>
      <instanceId>i-2ba64342</instanceId>
      <imageId>ami-60a54009</imageId>
      <instanceState>
        <code>0</code>
        <name>pending</name>
      </instanceState>
      <dnsName></dnsName>
      <keyName>example-key-name</keyName>
    </item>
    <item>
      <instanceId>i-2bc64242</instanceId>
      <imageId>ami-60a54009</imageId>
      <instanceState>
        <code>0</code>
        <name>pending</name>
      </instanceState>
      <dnsName></dnsName>
      <keyName>example-key-name</keyName>
    </item>
    <item>
      <instanceId>i-2be64332</instanceId>
      <imageId>ami-60a54009</imageId>
      <instanceState>
        <code>0</code>
        <name>pending</name>
      </instanceState>
      <dnsName></dnsName>
      <keyName>example-key-name</keyName>
    </item>
  </instancesSet>
</RunInstancesResponse>
```

Related Operations

- DescribeInstances
- TerminateInstances
- AuthorizeSecurityGroupIngress
- RevokeSecurityGroupIngress
- DescribeSecurityGroups

DescribeInstances

The `DescribeInstances` operation returns information about instances owned by the user making the request.

An optional list of instance IDs may be provided to request information for those instances only. If no instance IDs are provided, information of all relevant instances information will be returned. If an instance is specified that does not exist a fault is returned. If an instance is specified that exists but is not owned by the user making the request, then that instance will not be included in the returned results.

Recently terminated instances will be included in the returned results for a small interval subsequent to their termination. This interval is typically of the order of one hour.

Request Parameters

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

Element Name	Definition	Required?	Type
`instancesSet`	Set of instances IDs to get the status of.	Yes (but may be empty)	`xsd:string[]`

Response Tags

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

Element Name	Definition
`reservationSet`	A list of structures describing the status of all requested instances.

Sample Request

```
<DescribeInstances xmlns="http://ec2.amazonaws.com/doc/2006-06-26">
    <instancesSet>
        <item>
            <instanceId>i-28a64341</instanceId>
        </item>
    </instancesSet>
</DescribeInstances>
```

Sample Response

```
<DescribeInstancesResponse xmlns="http://ec2.amazonaws.com/doc/2006-06-26">
  <reservationSet>
    <item>
      <reservationId>r-44a5402d</reservationId>
      <ownerId>UYY3TLBUXIEON5NQVUUX6OMPWBZIQNFM</ownerId>
      <groupSet>
        <item>
          <groupId>default</groupId>
        </item>
      </groupSet>
      <instancesSet>
        <item>
          <instanceId>i-28a64341</instanceId>
          <imageId>ami-6ea54007</imageId>
          <instanceState>
            <code>0</code>
            <name>running</name>
          </instanceState>
          <dnsName>domU-12-31-33-00-00-5A.dc2.compute.amazonaws.com</dnsName>
          <keyName>example-key-name</keyName>
        </item>
      </instancesSet>
    </item>
  </reservationSet>
</DescribeInstancesResponse>
```

Related Operations

- RunInstances
- TerminateInstances

TerminateInstances

The `TerminateInstances` operation shuts down one or more instances. This operation is idempotent and terminating an instance that is in the process of shutting down (or already terminated) will succeed.

Terminated instances remain visible for a short period of time (approximately one hour) after termination, after which their instance ID is invalidated.

Request Parameters

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

Element Name	Definition	Required?	Type
instancesSet	One or more instance IDs returned from previous calls to `RunInstances`.	Yes	`xsd:string[]`

Response Tags

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

Element Name	Definition
instancesSet	A complex type containing describing the current and new state of each instance specified.

Sample Request

```
<TerminateInstances xmlns="http://ec2.amazonaws.com/doc/2006-06-26">
  <instancesSet>
    <item>
      <instanceId>i-28a64341</instanceId>
    </item>
  </instancesSet>
</TerminateInstances>
```

Sample Response

```
<TerminateInstancesResponse xmlns="http://ec2.amazonaws.com/doc/2006-06-26">
  <instancesSet>
    <item>
      <instanceId>i-28a64341</instanceId>
      <shutdownState>shutting-down</shutdownState>
      <previousState>running</previousState>
    </item></instancesSet>
</TerminateInstancesResponse>
```

Related Operations

- DescribeInstances

CreateSecurityGroup

The `CreateSecurityGroup` operation creates a new security group.

Every instance is launched in a security group. If none is specified as part of the launch request then instances are launched in the default security group. Instances within the same security group have unrestricted network access to one another. Instances will reject network access attempts from other instances in a different security group. As the owner of instances you may grant or revoke specific permissions using the AuthorizeSecurityGroupIngress and RevokeSecurityGroupIngress operations.

Request Parameters

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

Element Name	Definition	Required?	Type
`groupName`	Name for the new security group.	Yes	`xsd:string`
`groupDescription`	Description of the new security group.	Yes	`xsd:string`

Response Tags

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

Element Name	Definition
`return`	`true` if call succeeded.

Sample Request

```
<CreateSecurityGroup xmlns="http://ec2.amazonaws.com/doc/2006-06-26">
    <groupName>WebServers</groupName>
    <groupDescription>Web</groupDescription>
</CreateSecurityGroup>
```

Sample Response

```
<CreateSecurityGroupResponse xmlns="">
  <return>true</return>
</CreateSecurityGroupResponse>
```

Related Operations

- RunInstances
- DescribeSecurityGroups
- AuthorizeSecurityGroupIngress
- RevokeSecurityGroupIngress
- DeleteSecurityGroup

DescribeSecurityGroups

The `DescribeSecurityGroups` operation returns information about security groups owned by the user making the request.

An optional list of security group names may be provided to request information for those security groups only. If no security group names are provided, information of all security groups will be returned. If a group is specified that does not exist a fault is returned.

Request Parameters

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

Element Name	Definition	Required?	Type
securityGroupSet	List of security groups to describe.	Yes	`xsd:string[]`

Response Tags

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

Element Name	Definition
securityGroupInfo	Information about security groups.

Sample Request

```
<DescribeSecurityGroups xmlns="http://ec2.amazonaws.com/doc/2006-06-26">
  <securityGroupSet>
    <item>
      <groupName>WebServers</groupName>
    </item>
    <item>
      <groupName>RangedPortsBySource</groupName>
    </item>
  </securityGroupSet>
</DescribeSecurityGroups>
```

Sample Response

```
<DescribeSecurityGroupsResponse xm-
lns="http://ec2.amazonaws.com/doc/2006-06-26">
  <securityGroupInfo>
    <item>
      <ownerId>UYY3TLBUXIEON5NQVUUX6OMPWBZIQNFM</ownerId>
      <groupName>WebServers</groupName>
      <groupDescription>Web</groupDescription>
      <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>
    </item>
    <item>
```

```
      <ownerId>UYY3TLBUXIEON5NQVUUX6OMPWBZIQNFM</ownerId>
      <groupName>RangedPortsBySource</groupName>
      <groupDescription>A</groupDescription>
      <ipPermissions>
         <item>
          <ipProtocol>tcp</ipProtocol>
          <fromPort>6000</fromPort>
          <toPort>7000</toPort>
          <groups/>
          <ipRanges/>
        </item>
      </ipPermissions>
    </item>
  </securityGroupInfo>
</DescribeSecurityGroupsResponse>
```

Related Operations

- CreateSecurityGroup
- AuthorizeSecurityGroupIngress
- RevokeSecurityGroupIngress
- DeleteSecurityGroup

DeleteSecurityGroup

The `DeleteSecurityGroup` operation deletes a security group.

If an attempt is made to delete a security group and any instances exist that are members of that group a fault is returned.

Request Parameters

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

Element Name	Definition	Required?	Type
groupName	Name of the security group to delete.	Yes	`xsd:string`

Response Tags

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

Element Name	Definition
return	`true` if group deleted.

Sample Request

```
<DeleteSecurityGroup xmlns="http://ec2.amazonaws.com/doc/2006-06-26">
    <groupName>RangedPortsBySource</groupName>
</DeleteSecurityGroup>
```

Sample Response

```
<DeleteSecurityGroupResponse xmlns="http://ec2.amazonaws.com/doc/2006-06-26">
  <return>true</return>
</DeleteSecurityGroupResponse>
```

Related Operations

- CreateSecurityGroup
- DescribeSecurityGroups
- AuthorizeSecurityGroupIngress
- RevokeSecurityGroupIngress

AuthorizeSecurityGroupIngress

The `AuthorizeSecurityGroupIngress` operation adds permissions to a security group.

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

Note
Changes are anticipated in this API that may restrict further what is allowable. Please consult the section called “Anticipated API changes” for more details.

Permission changes are propagated to instances within the security group being modified as quickly as possible. However, a small delay is likely, depending on the number of instances that are members of the indicated group.

Request Parameters

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

Element Name	Definition	Re-quired?	Type
userId	AWS Access Key ID.	Yes	`xsd:string`
groupName	Name of the group to modify.	Yes	`xsd:string`
ipPermissions	Set of permissions to add to the group.	Yes	`ec2:IpPermissionType[]`

Response Tags

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

Element Name	Definition
return	`true` if permissions successfully added.

Sample Request

```
<AuthorizeSecurityGroupIngress xm-
lns="http://ec2.amazonaws.com/doc/2006-06-26">
    <userId/>
    <groupName>WebServers</groupName>
    <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>
```

```
<AuthorizeSecurityGroupIngress xm-
lns="http://ec2.amazonaws.com/doc/2006-06-26">
    <userId/>
    <groupName>RangedPortsBySource</groupName>
    <ipPermissions>
        <item>
            <ipProtocol>tcp</ipProtocol>
            <fromPort>6000</fromPort>
            <toPort>7000</toPort>
            <groups/>
            <ipRanges/>
            <dnsNames>
                <item>
                    <dnsName>host.example.com</dnsName>
                </item>
            </dnsNames>
        </item>
    </ipPermissions>
</AuthorizeSecurityGroupIngress>
```

Sample Response

```
<AuthorizeSecurityGroupIngressResponse xm-
lns="http://ec2.amazonaws.com/doc/2006-06-26">
  <return>true</return>
</AuthorizeSecurityGroupIngressResponse>
```

Related Operations

- CreateSecurityGroup
- DescribeSecurityGroups
- RevokeSecurityGroupIngress
- DeleteSecurityGroup

RevokeSecurityGroupIngress

The `RevokeSecurityGroupIngress` operation revokes existing permissions that were previously granted to a security group. The permissions to revoke must be specified using the same values originally used to grant the permission.

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

Note
Changes are anticipated in this API that may restrict further what is allowable. Please consult the section called "Anticipated API changes" for more details.

Permission changes are propagated to instances within the security group being modified as quickly as possible. However, a small delay is likely, depending on the number of instances that are members of the indicated group.

Request Parameters

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

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

Response Tags

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

Element Name	Definition
return	`true` if permissions successfully revoked.

Sample Request

```
<RevokeSecurityGroupIngress xmlns="http://ec2.amazonaws.com/doc/2006-06-26">
    <userId/>
    <groupName>RangedPortsBySource</groupName>
    <ipPermissions>
        <item>
            <ipProtocol>tcp</ipProtocol>
            <fromPort>6000</fromPort>
            <toPort>7000</toPort>
            <groups/>
            <ipRanges/>
        </item>
    </ipPermissions>
</RevokeSecurityGroupIngress>
```

Sample Response

```
<RevokeSecurityGroupIngressResponse xm-
lns="http://ec2.amazonaws.com/doc/2006-06-26">
  <return>true</return>
</RevokeSecurityGroupIngressResponse>
```

Related Operations

- CreateSecurityGroup
- DescribeSecurityGroups
- AuthorizeSecurityGroupIngress
- DeleteSecurityGroup

Data Types

The Amazon EC2 API contains several data types used by the various operations. This section describes each operation in detail.

DescribeKeyPairsResponseItemType

The *`DescribeKeyPairsResponseItemType`* data type.

Relevant Operations

Operations that use this data type include:

- `DeleteKeypair`
- `DescribeKeypairs`

Contents

The following table describes and shows the elements contained in DescribeKeyPairsResponseItemType.

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

DescribeImagesResponseItemType

The *`DescribeImagesResponseItemType`* data type.

Relevant Operations

Operations that use this data type include:

- `DescribeImages`

Contents

The following table describes and shows the elements contained in DescribeImagesResponseItemType.

Member	Description	Type
`imageId`	Unique ID of the AMI being described.	`xsd:string`
`imageState`	Current state of the AMI.	`xsd:string`

Member	Description	Type
	• `available`: the image has been successfully registered and is available for launching • `deregistered`: the image has recently been deregistered and is no longer available for launching	
`imageOwnerId`	AWS Access Key ID of the image owner.	`xsd:string`
`isPublic`	`true` if anyone other than the owner can launch instances of this image.	`xsd:boolean`

IpPermissionType

The *`IpPermissionType`* data type.

Relevant Operations

Operations that use this data type include:

- AuthorizeSecurityGroupIngress
- DescribeSecurityGroups
- RevokeSecurityGroupIngress

Contents

The following table describes and shows the elements contained in IpPermissionType.

Member	Description	Type
`ipProtocol`	IP Protocol.	`xsd:string`
`fromPort`	Start of port range.	`xsd:int`
`toPort`	End of port range.	`xsd:int`
`groups`	List of security group and user ID pairs.	`ec2:UserIdGroupPairType[]`
`ipRanges`	List of CIDR IP range specifications.	`xsd:string[]`

UserIdGroupPairType

The *`UserIdGroupPairType`* data type.

Relevant Operations

Operations that use this data type include:

- AuthorizeSecurityGroupIngress
- DescribeSecurityGroups
- RevokeSecurityGroupIngress

Contents

The following table describes and shows the elements contained in UserIdGroupPairType.

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

SecurityGroupItemType

The *`SecurityGroupItemType`* data type.

Relevant Operations

Operations that use this data type include:

- DescribeSecurityGroups

Contents

The following table describes and shows the elements contained in SecurityGroupItemType.

Member	Description	Type
`ownerId`	AWS Access Key ID of the owner of the security group described.	`xsd:string`
`groupName`	Name of the security group.	`xsd:string`
`groupDescription`	Description of the security group.	`xsd:string`
`ipPermissions`	Set of IP permissions associated with the security group.	`ec2:IpPermissionType[]`

RunInstanceItemType

The *`RunInstanceItemType`* data type.

Relevant Operations

Operations that use this data type include:

- RunInstances

Contents

The following table describes and shows the elements contained in RunInstanceItemType.

Member	Description	Type
imageId	Unique ID of a machine image, returned by a call to RegisterImage.	xsd:string
minCount	Minimum number of instances to launch. If *minCount* is more than Amazon EC2 can launch, no instances are launched at all.	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.	xsd:int
keyName	The name of the keypair.	xsd:string

InstanceStateType

The *InstanceStateType* data type.

Relevant Operations

Operations that use this data type include:

- RunInstances
- DescribeInstances
- TerminateInstances

Contents

The following table describes and shows the elements contained in InstanceStateType.

Member	Description	Type
code	A 16 bit unsigned integer. The high byte is an opaque internal value and should be ignored when consulting this value. The low byte is set based on the state represented: • pending: 0 • running: 16 • shutting-down: 32 • terminated: 48	xsd:int
name	The current state of the instance.	xsd:string

Member	Description	Type
	• `pending`: the instance is in the process of being launched • `running`: the instance has been launched (although it may not yet have completed the boot process) • `shutting-down`: the instance has begun the shutdown process • `terminated`: the instance has been terminated	

GroupSetType

The *`GroupSetType`* data type.

Relevant Operations

Operations that use this data type include:

- RunInstances

Contents

The following table describes and shows the elements contained in GroupSetType.

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

RunningInstancesItemType

The *`RunningInstancesItemType`* data type.

Relevant Operations

Operations that use this data type include:

- RunInstances

Contents

The following table describes and shows the elements contained in `RunningInstancesItemType`.

Element Name	Description	Type
`instanceId`	Unique ID of the instance launched.	`xsd:string`
`imageId`	Image ID of the AMI used to launch	`xsd:string`

Element Name	Description	Type
	the instance.	
instanceState	The current state of the instance. • `pending`: the instance is in the process of being launched • `running`: the instance has been launched (although it may not yet have completed the boot process) • `shutting-down`: the instance has begun the shutdown process • `terminated`: the instance has been terminated	`ec2:InstanceStateType`
dnsName	The DNS name assigned to the instance. This element remains empty until the instance enters a running state.	`xsd:string`
reason	An optional reason for the most recent state transition. This may be an empty string.	`xsd:string`
keyName	An optional key name. If this instance was launched with an associated key pair, this is the name of that key pair.	`xsd:string`

ReservationInfoType

The *`ReservationInfoType`* data type.

Relevant Operations

Operations that use this data type include:

- RunInstances
- DescribeInstances

Contents

The following table describes and shows the elements contained in ReservationInfoType.

Member	Description	Type
reservationId	Unique ID of the reservation being described.	`xsd:string`
ownerId	AWS Access Key ID of the user who owns the reservation.	`xsd:string`

Member	Description	Type
groupSet	Set of security groups these instances were launched in.	ec2:GroupSetType[]
instancesSet	Information about instances started.	ec2:RunningInstancesItemType[]

TerminateInstancesResponseInfoType

The *TerminateInstancesResponseInfoType* data type.

Relevant Operations

Operations that use this data type include:

- TerminateInstances

Contents

The following table describes and shows the elements contained in TerminateInstancesResponseInfoType.

Element Name	Description	Type
instanceId	Instance ID returned from previous call to RunInstances.	xsd:string

Technical FAQ

1. Why can't I "talk" to my instances?

   Here are a few common reasons for broken connectivity to your instance.

   An instance's state is changed to running as soon as we start to boot your OS. This means there will be some delay (possibly a few minutes depending on your configuration) during which your instance will not have been fully set-up. After this period, it should be fully functional.

   Additionally, you will need to make sure you have 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 by default and disallow all access from the Internet and other groups. You will need to add permissions to your `default` group or you will have to set up a new group with appropriate permissions. See the developer guide for more information on the “Securing the Network”.

   Assuming you have authorized port 22, a useful debugging tool is to try to open an ssh connection with verbose output. You should use the man page to get the exact syntax for your system, but the command is likely to look like `ssh -vv root@[hostname]`. This output would be very useful if posting to the forum.

2. Why did my instance terminate immediately after launch?

   Launch errors may be the result of an internal error during launch or a corrupt Amazon EC2 image. The former should be rare, and we actively test for and isolate suspect hosts. You should use the “DescribeInstances” API to look for more details on why your instance failed to launch.

   NB: the `ec2-describe-instances` command line tool does not conveniently print out this information yet! You can use the `-v` flag to read the SOAP response from this tool and get the information discussed above.

   You can always feel free to attempt to launch the image again, but if you run into a persistent problem (especially with a shared image), you should post to the Amazon EC2 forum.

3. 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. What's happening?

   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 shut the instance down using the TerminateInstances call (`ec2-terminate` on the command line).

4. What username do I use for the various Amazon EC2 tools?

   When you sign up with Amazon Web Services, you are given an AWS Account ID. This is your username. More detail is provided in the Getting Started Guide.

5.

What happens to my running instances if the machines they are running on go down?

The instances themselves will be terminated and will have to be relaunched. The data on the instances' hard drives will be lost.

Always replicate important data or store it in Amazon S3.

6.

Why do my instances take so long to start?

Amazon EC2 has to move the images around the network before they can be launched. For big images and/or congested networks, this can take several minutes. Images are cached to alleviate this problem, so it should be less noticeable as you use your images more frequently.

7.

Why are my instances stuck in a pending state (or a shutting-down state)?

This situation should be rare and is the result of a software error or misconfiguration. We actively monitor for it, but please let us know if you do encounter this.

8.

I get a "permission denied, wrong user" 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 need to use the same ID and certificate to register the image with Amazon EC2.

9.

I get an "InsufficientInstanceCapacity" error when I try to launch an instance.

This error indicates that we don’t currently have enough available capacity to service your request. During our beta, capacity is limited.

If you are requesting a large number of instances, there may not be enough server capacity to host them. You could try again at a different time or specify a smaller number of instances to launch.

10.

I get an "InstanceLimitExceeded" error when I try to launch an instance.

This error indicates that you have reached your concurrent running instance limit. For new users during the public beta, this limit is 20.

If you need additional capacity, please contact us at aws@amazon.com.

11.

How many instances can I launch?

Each user has a concurrent running instance limit. For new users during the public beta, this limit is 20.

12.

Can I use a static IP in my instances?

No. Your image must be configured as a DHCP client and it will be assigned an IP.

13.

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.

14. How do I handle time synchronization between instances?

You can set up NTP (the Network Time Protocol) which does this for you. You can find more information at http://-www.ntp.org/. This is particularly important if you plan on using any of Amazon's web services (such as Amazon S3 or Amazon EC2) from within an instance, since requests to these services need to be timestamped.

15. Can I use my own kernel?

Not at present.

16. Can I get a bigger/smaller/differently optimized virtual machine?

Not at present. For now, if you need more capacity launch more instances.

17. Is there a REST interface to Amazon EC2?

Not at present. For now, you will have to use the SOAP API or the provided API command line tools.

18. How does Amazon EC2 handle load balancing?

With a service as flexible as Amazon EC2, customers can launch any number of load balancing systems within Amazon EC2. The load balancing instances can forward traffic to other systems. There are several open source solutions that are in wide use.

19. How do I monitor my systems?

Amazon EC2 currently only provides the most basic monitoring. You can tell from DescribeInstances whether we believe your instance is running or not. However, you may regard your systems running in Amazon EC2 as your data center, and so any monitoring instrumentation that you wish to include on the systems – be it SNMP or some other mechanism – is entirely up to you.

20. Is there any way for an instance to discover its own instance ID?

Not at present. For now you can get the instance ID by bundling an image with your credentials and calling **ec2-describe-instances**. The return values contain both the hostname (which the instance knows) and the instance ID.

The following command should do the trick (assuming the command line tools have been configured correctly): **`ec2-describe-instances | grep $HOSTNAME | awk '{print $2}'`**

21. Can I pass arbitrary configuration values to an instance at launch time?

    Not at present.

22. Is there a way to run a script on instance termination?

    Not with any reliability. Amazon EC2 tries to shut an instance down cleanly (in which case normal system shutdown scripts will run), but there is only a short time available for things to happen and in some cases (hardware failure, for example) this does not happen. Since there is no entirely reliable way to ensure shutdown scripts run, it is best to have a strategy in place to deal with abnormal terminations.

23. Why do I get keep getting `"Request has expired"` errors?

    To reduce the risk of replay attacks our requests include a timestamp. This, along with the most important parts of the request, is signed to ensure the message (including the timestamp) can't 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 deemed 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. NTP is a good way to do this.

Glossary

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.
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	A set of customer instances that have been designated by the customer as being related by assigning them the same security group when the instances were first run. The Amazon EC2 firewall controls access to instances based on the instance's group membership and the rules defined for the group.
Reservation	A collection if instances started as part of the same launch request.