

# **Amazon Elastic Compute Cloud**

**Developer Guide**

**API Version 2007-08-29**



## Amazon Elastic Compute Cloud: Developer Guide

Copyright © 2007 Amazon Web Services LLC or its affiliates. All rights reserved.

## Table of Contents

What's New .....	1
Welcome .....	2
Introduction to Amazon Elastic Compute Cloud .....	5
Creating and Preparing AMIs .....	6
Creating an AMI .....	6
Starting with an Existing AMI .....	6
Creating an AMI through a Loopback File .....	12
Bundling an AMI .....	18
Sharing AMIs .....	21
Shared AMI Guidelines .....	21
Sharing AMIs .....	25
Making an AMI Public .....	25
Sharing an AMI with Specific Users .....	26
Publishing Shared AMIs .....	27
Creating Paid AMIs .....	28
Paid and Supported AMIs .....	29
Summary of How Paid AMIs Work .....	29
Summary of How Supported AMIs Work .....	30
Product Registration .....	31
Associating a Product Code with an AMI .....	32
Sharing Your Paid AMI with Select Users or the Public .....	33
Confirming an Instance Is Running an AMI Associated with a Product Code .....	33
Getting the Product Code from Within an Instance .....	34
Launching and Using Instances .....	35
Instance Usage .....	35
Instance Types .....	36
Instance Metadata .....	37
Instance Storage .....	43
Using Shared AMIs .....	45
Paying for AMIs .....	46
Get Console Output and Reboot Instances .....	49
Instance Addressing and Network Security .....	50
Instance Addressing .....	50
Network Security .....	52
Concepts .....	52
Examples .....	53
Using the APIs .....	56
Using the SOAP API .....	56
Using the Query API .....	59
API Reference .....	63
API Conventions .....	63
API Versioning .....	64
API Error Codes .....	65
Common Data Types .....	67
DescribeImagesResponseType .....	67
DescribeKeyPairsResponseType .....	68
EmptyElementType .....	69
GroupSetType .....	69
InstanceStateType .....	69
IpPermissionType .....	70
LaunchPermissionItemType .....	71
LaunchPermissionOperationType .....	71
ProductCodeItemType .....	72

---

ProductInstanceResponseItemType .....	72
ReservationInfoType .....	73
RunInstanceItemType .....	73
RunningInstancesItemType .....	74
SecurityGroupItemType .....	75
TerminateInstancesResponseInfoType .....	76
UserDataType .....	76
UserIdGroupPairType .....	77
Amazon EC2 SOAP API .....	78
List of Operations by Function .....	78
AuthorizeSecurityGroupIngress .....	79
ConfirmProductInstance .....	80
CreateKeyPair .....	81
CreateSecurityGroup .....	82
DeleteKeyPair .....	84
DeleteSecurityGroup .....	84
DeregisterImage .....	85
DescribeImageAttribute .....	86
DescribeImages .....	88
DescribeInstances .....	90
DescribeKeyPairs .....	91
DescribeSecurityGroups .....	93
GetConsoleOutput .....	94
ModifyImageAttribute .....	95
RebootInstances .....	97
RegisterImage .....	97
ResetImageAttribute .....	98
RevokeSecurityGroupIngress .....	99
RunInstances .....	101
TerminateInstances .....	104
Amazon EC2 Query API .....	105
Common Query Parameters .....	105
List of Operations by Function .....	106
AuthorizeSecurityGroupIngress .....	107
ConfirmProductInstance .....	109
CreateKeyPair .....	110
CreateSecurityGroup .....	112
DeleteKeyPair .....	113
DeleteSecurityGroup .....	113
DeregisterImage .....	115
DescribeImageAttribute .....	115
DescribeImages .....	117
DescribeInstances .....	119
DescribeKeyPairs .....	121
DescribeSecurityGroups .....	122
GetConsoleOutput .....	123
ModifyImageAttribute .....	124
RebootInstances .....	126
RegisterImage .....	127
ResetImageAttribute .....	128
RevokeSecurityGroupIngress .....	129
RunInstances .....	131
TerminateInstances .....	134
Command Line Tools Reference .....	136
AMI Tools .....	137
ec2-bundle-image .....	137
ec2-bundle-vol .....	139
ec2-delete-bundle .....	142

---

ec2-download-bundle .....	144
ec2-unbundle .....	146
ec2-upload-bundle .....	147
API Tools .....	149
List of Operations by Function .....	149
ec2-add-group .....	150
ec2-add-keypair .....	151
ec2-authorize .....	152
ec2-confirm-product-instance .....	154
ec2-delete-group .....	155
ec2-delete-keypair .....	156
ec2-deregister .....	157
ec2-describe-groups .....	158
ec2-describe-image-attribute .....	159
ec2-describe-images .....	160
ec2-describe-instances .....	162
ec2-describe-keypairs .....	163
ec2-fingerprint-key .....	164
ec2-get-console-output .....	165
ec2-modify-image-attribute .....	165
ec2-reboot-instances .....	168
ec2-register .....	168
ec2-reset-image-attribute .....	169
ec2-revoke .....	170
ec2-run-instances .....	172
ec2-terminate-instances .....	174
Technical FAQ .....	176
General Information .....	176
Operation Information .....	177
Instance Types and Architectures .....	179
IP Information .....	180
Monitoring, Errors, and Unexpected Behavior .....	181
Error Messages .....	182
Paid AMIs .....	183
Miscellaneous .....	184
Glossary .....	185
Document Conventions .....	186

---

# What's New

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

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

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

# Welcome

## Topics

- [Audience](#)
- [How This Guide Is Organized](#)
- [Related Resources](#)

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

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

## Audience

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

## Required Knowledge and Skills

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

- XML (See [W3 Schools XML Tutorial](#))
- Basic understanding of web services (See [W3 Schools Web Services Tutorial](#))
- A programming language for consuming a web service and any related tools

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

## How This Guide Is Organized

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

Information	Relevant Sections
Describes how to create a customized software package (operating system and applications) that you can run on Amazon EC2	<ul style="list-style-type: none"><li>• <a href="#">Creating and Preparing AMIs</a></li></ul>
Describes Amazon EC2 instances and provides tips for using them effectively	<ul style="list-style-type: none"><li>• <a href="#">Launching and Using Instances</a></li></ul>
Describes instance network addressing, explains the distributed firewall, and provides usage examples	<ul style="list-style-type: none"><li>• <a href="#">Instance Addressing and Network Security</a></li></ul>
Explains the basics of using the SOAP and Query APIs, including signing requests	<ul style="list-style-type: none"><li>• <a href="#">Using the APIs</a></li></ul>
Provides a comprehensive reference to the SOAP and Query APIs	<ul style="list-style-type: none"><li>• <a href="#">API Reference</a></li></ul>
Provides a comprehensive reference to the Amazon EC2 command line tools	<ul style="list-style-type: none"><li>• <a href="#">Command Line Tools Reference</a></li></ul>
Provides answers to commonly asked questions	<ul style="list-style-type: none"><li>• <a href="#">Technical FAQ</a></li></ul>
Describes Amazon EC2 terms	<ul style="list-style-type: none"><li>• <a href="#">Glossary</a></li></ul>
Typographic and symbol conventions	<ul style="list-style-type: none"><li>• <a href="#">Document Conventions</a></li></ul>

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

## Related Resources

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

Resource	Description
<a href="#">Amazon EC2 Getting Started Guide</a>	The Getting Started Guide provides a quick tutorial of the service based on a simple use case. Examples and instructions are included.
<a href="#">Amazon EC2 Release Notes</a>	The Release Notes give a high-level overview of the current



## Amazon Elastic Compute Cloud Developer Guide

### Related Resources

---

Resource	Description
	release. They specifically note any new features, corrections, and known issues.
<a href="#">AWS Developer Resource Center</a>	A central starting point to find documentation, code samples, release notes, and other information to help you build innovative applications with AWS.
<a href="#">Amazon EC2 product information</a>	The primary web page for information about Amazon EC2.
<a href="#">Discussion Forums</a>	A community-based forum for developers to discuss technical questions related to Amazon Web Services.
E-mail address for questions related to your AWS account: <webservices@amazon.com>	This e-mail address is <i>only</i> for account questions. For technical questions, use the Discussion Forums.
<a href="#">Conditions of Use</a>	Detailed information about the copyright and trademark usage at Amazon.com.

# Introduction to Amazon Elastic Compute Cloud

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

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

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

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

# Creating and Preparing AMIs

## Topics

- [Creating an AMI](#)
- [Bundling an AMI](#)
- [Building Shared AMIs](#)
- [Sharing AMIs](#)
- [Paid AMIs](#)

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

## Creating an AMI

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

The easiest method involves starting from an existing public AMI and modifying it according to your requirements, as described in [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 [Creating an AMI through a Loopback File](#).

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

## Starting with an Existing AMI

To quickly and easily get a new working AMI, start with an existing public AMI or one of your own. You can then modify it and create a new AMI with the `ec2-bundle-vol` utility described in [Bundling an AMI](#).



### Note

Before selecting an AMI, determine whether the instance types you plan to launch are 32-bit or 64-bit. For more information, see [Instance Types](#)

Make sure you are using GNU Tar 1.15 or later.

To use an existing AMI to create a new AMI:

1. [Select an AMI](#)
2. [Generate a Key Pair](#)
3. [Launch the Instance](#)
4. [Authorize Network Access](#)
5. [Connect to the Instance](#)
6. [Upload the Key and Certificate](#)

## Select an AMI

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

**To select an AMI:**

1. Get a list of available AMIs by entering the `ec2-describe-images` command:

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

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

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

### Example

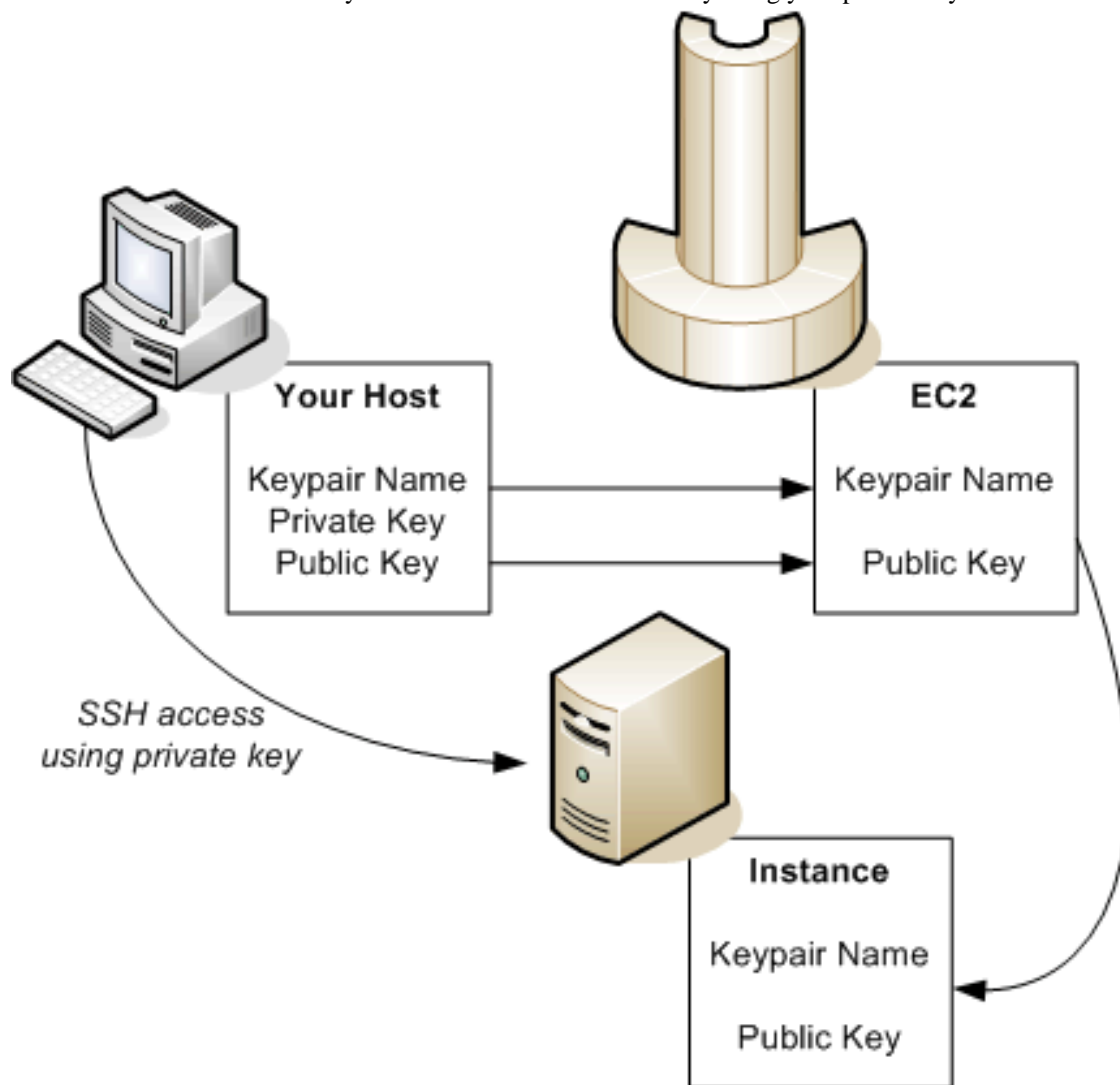
```
PROMPT> ec2-describe-images -o self -o amazon
IMAGE ami-60a54009 ec2-public-images/base-fc4-apache.manifest.xml
475219833042 available public
IMAGE ami-61a54028 <your-s3-bucket>/image.manifest.xml 495219933132 available
private
IMAGE ami-2bb65342 ec2-public-images/getting-started.manifest.xml
475219833042 available public
IMAGE ami-6ea54007 ec2-public-images/base-fc3-mysql.manifest.xml 475219833042
available public
```

## Generate a Key Pair

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

After you generate a key pair, the public key is stored in Amazon EC2 using the key pair name you

selected. Whenever you launch an instance using the key pair name, the public key is copied to the instance metadata. This allows you to access the instance securely using your private key.



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

1. Enter the following command:

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

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

The resulting private key is displayed.

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



#### Note

This file should only be readable by the file owner.

#### Example

```
PROMPT> ec2-add-keypair gsg-keypair
KEYPAIR gsg-keypair
1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:f1:6f
-----BEGIN RSA PRIVATE KEY-----
MIIEoQIBAAKCAQBULFg5ujHrtmljnutSuo08Xe56LlT+HM8v/xkaa39EstM3/aFXTHgElQiJLChp
HungXQ29VTc8rc1bw0lkdi23OH5eqkMHGhvEwqa0HWASUM1l4o3o/IX+0f2UcPoKCOVUR+jx71Sg
5AU52EQfanIn3ZQ8lFW7Edp5a3q4DhjG1UKToHVbicL5E+g45zfB95wIyywWZfeW/UUF3LpGZyq/
ebIUlq1qTbHkLbCC2r7RTn8vpQWp47BGVYGtGSBmPTrP5hnbzzuqj3itkiLHjU39S2sJCJ0TrJx5
i8BygR4s3mHKBj8l+ePQxG1kGbF6R4yg6sECmXn17MRQVXODNHZbAgMBAAECggEAY1tsiUsIwD15
91CXirkYGuvfLyLflXenxfI50mDFms/mumTqloHO7tr0oriHDR5K7wMcY/YY5YkcXNo7mvUVD1pM
ZNUJs7rw9gzRTTrf7LylaJ58kOcyajw8TsC4e4LPbFaHwSld6K8rXh64o6WgW4SrsB6ICmrlkGQI7
3wcfgt5ecIu4Tzf00E9IHjn+2eRlsrjBdeORI7KiUNC/pAG23I6MdDOFEQRcCSigCj+4/mciFUSA
SWS4dMbrpb9FNSIcf9dcLxVM7/6KxgJNfZc9XWzUw77Jg8x92Zd0fVhHouX5IZC+UvSKWB4dyfcI
tE8C3p9bbU9VGyY5vLCAiIb4qQKBgQDLiO24GXrIkswF32YtBBMuVgLGcWU9h9H1O9mKAc2m8Cm1
jUE5IpzRjTedc9I2qiIMUTwtgnw42auSCzbUeYMURPtDqyQ7p6A jMu jp9EPemcSVOK9vXYL0Ptco
xW9MC0dtV6iPkCN7gOqiZXPRKaFbWADp16p8UAIvS/a5XXk5jwKBgQCKkpHi2EIShluRkxhljyWC
iDCiK6JBRsMvpLbc0v5dKwP5alo1fmdR5PJaV2qvZSj5CYNpMay1/EDNTY50SIJU+0KFmQbyhsbm
rdLNLDL4+TcnT7c62/aH0lohYaf/VCbRhtLlBfqGoQc7+sAc8vmKkesnF7CqCEKDYf/dhrxYdQKB
gC0iZzzNAapayz1+JcVTWwEid6j9JqNXbBc+Z2YwMi+T0Fv/P/hwkX/ypeOXnIUcw0Ih/YtGBVAC
DQbsz7LcY1HqXiHKYNWNvXgwwO+oiChjxvEkSdsTTIfnK4VSCvU9BxDbQHjdiNDJbL6oar92UN7V
rBYvChJZF7LvUH4YmVpHAoGAbZ2X7XvoveEO+uZ58/BGKOIGHByHBDiXtzmHdJr15HTYjxK7OgTZm
gK+8zp4L9IbvLGDMJO8vft32XPEWuvI8twCzFH+CsWLQADZMZKSsBasOZ/h1FwhdMgCMcY+Qlzd4
JZKjTSu3i7vhvx6RzdSedXEMNTZWN4qlIx3kR5aHcukCgYA9T+ZrvmlF0seQPbLknn7EqhXIjBaT
P8TTvW/6bdPi23ExzxZn7K0drfclYRphlLHMPaONv/x2xALIf91UB+v5ohyloDoasL0gi j1houRe
2ERKKdwz0ZL9SWq6VTdhr/5G994CK72fy5WhyERbDJUIdHaK3M849JJuF8cSrvSb4g==
-----END RSA PRIVATE KEY-----
```

## Launch the Instance

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

#### To launch an instance

1. Start the launch by entering the following command:

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

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

Launching the instance will take a few minutes.

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

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

where `<instance_id>` is the ID of the instance.

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



### Important

Once you launch an instance, you will be billed for all usage, including hourly CPU time. Make sure to terminate any instances that you do not want to leave running. For information on Amazon EC2 pricing, refer to the [Amazon EC2 home page](#).

### Example

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

```
PROMPT> ec2-run-instances ami-2bb65342 -k gsg-keypair
RESERVATION    r-302dc059    416161254515    default
INSTANCE       i-eb977f82    ami-2bb65342
keypair        0            m1.small        2007-10-16T07:56:20+0000    pending gsg-
```

The following shows the status of the launch:

```
PROMPT> ec2-describe-instances i-eb977f82
RESERVATION    r-302dc059    416161254515    default
INSTANCE       i-eb977f82    ami-2bb65342
ec2-72-44-40-222.compute-1.amazonaws.com    domU-
12-31-35-00-10-74.compute-1.internal    running gsg-keypair    0
m1.small        2007-10-16T07:56:20+0000
```

## Authorize Network Access

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

### To enable SSH service on port 22

- Enter the following command:

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

## Connect to the Instance

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

### To connect to an instance

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

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

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

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



### Important

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

### Example

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

```
PROMPT> ssh -i id_rsa-gsg-keypair root@domU-  
12-34-31-00-00-05.usm1.compute.amazonaws.com  
root@domU-12-34-31-00-00-05 #
```

## Upload the Key and Certificate

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



### Note

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

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

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

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

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

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





#### Note

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

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

#### Example

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

## Creating an AMI through a Loopback File

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



#### Note

Before selecting an AMI, determine whether the instance types you plan to launch are 32-bit or 64-bit. For more information, see [Instance Types](#)

Make sure you are using GNU Tar 1.15 or later.

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

#### AMI Creation Process

1	<a href="#">Create a File to Host the AMI</a>
2	<a href="#">Create a Root File System Inside the File</a>
3	<a href="#">Mount the File through Loopback</a>
4	<a href="#">Prepare for the Installation</a>
5	<a href="#">Install the Operating System</a>
6	<a href="#">Configure the Operating System</a>

## Create a File to Host the AMI

The `dd` utility can create files of arbitrary sizes. Make sure to create a file large enough to host the

operating system, tools, and applications that you will install. For example, a baseline Linux installation requires about 700MB, so your file should be at least 1 GB.

### To create a file to host the AMI

- Enter the following command:

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

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

### Example

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

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

## Create a Root File System Inside the File

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

### To create an `ext3` file system

- Enter the following command:

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

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

### Example

The following command creates an `ext3` file system:

```
# mke2fs -F -j my-image.fs
mke2fs 1.38 (30-Jun-2005)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
131072 inodes, 262144 blocks
13107 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=268435456
8 block groups
32768 blocks per group, 32768 fragments per group
16384 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376

Writing inode tables: done
```

```
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 24 mounts or
180 days, whichever comes first. Use tune2fs -c or -i to override.
```

## Mount the File through Loopback

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

### To mount the file through loopback

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

```
# mkdir <image_mountpoint>
```

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

2. Mount the file system image:

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

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

### Example

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

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

## Prepare for the Installation

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

### To prepare for the installation

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

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

where `<image_mountpoint>` is the mount location.

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

```
/dev/sda1  /          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
```

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

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

[base]
name=Fedora Core 4 - $basearch - Base
mirrorl
ist=http://fedora.redhat.com/download/mirrors/fedora-core-$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
```

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

4. Enter the following:

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

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

## Example

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

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

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

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

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

## Install the Operating System

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

### To install the operating system

- Enter the following command:

```
# yum -c <yum_configuration_file> --installroot=<image_mountpoint> -y groupinstall Base
```

where `<yum_configuration_file>` is the name of the yum configuration file and `<image_mountpoint>` is the mount location.

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

### Example

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

```
# yum -c yum-xen.conf --installroot=/mnt/ec2-fs -y groupinstall Base
Setting up Group Process
Setting up repositories
base 100% |=====| 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!
```

## Configure the Operating System

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

## To configure the operating system

1. Edit (or create) `/mnt/ec2-fs/etc/sysconfig/network-scripts/ifcfg-eth0` and make 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. Additionally, the local hostname will be the same for all instances of the AMI, which might be confusing.

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

```
NETWORKING=yes
```

3. Amazon EC2 provides the instance with additional local disk storage on `/dev/sda2` and swap space on `/dev/sda3`. To ensure both these are mounted at system startup, add the following lines to `/mnt/ec2-fs/etc/fstab`:

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



### Note

The `/dev/sda2` and `/dev/sda3` storage locations only apply to small instances. For more information on instance storage, see [the section called “Instance Storage”](#).

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

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

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

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

where `<image_mountpoint>` is the mount location.

### Example

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

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

## Bundling an AMI

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

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

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

The AMI tools include three command-line utilities:

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

## Installing the AMI Tools

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

The AMI tools RPM is available from our [public Amazon S3 downloads bucket](#).

### To install the AMI tools

1. Install Ruby using the yum package manager:  

```
# yum install ruby
```
2. Install the AMI tools RPM:  

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

## Installation Issues

The AMI tools libraries install in `/usr/lib/site_ruby`.

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

## Viewing Documentation

### To view the manual for each utility

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

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

### To view help for each utility

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

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

## Bundling an AMI Using the AMI Tools

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

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

- Enter the following command:

```
# ec2-bundle-image -i <image_name>.img -k <private_keyfile> -c  
<certificate_file> -u <user_id>
```

where `<image_name>` is the name of the image file, `<private_keyfile>` is the file that contains the private key, `<certificate_file>` is the file that contains the certificate, and `<user_id>` is the user ID associated with your account.



#### Note

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

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

- Enter the following command:

```
# ec2-bundle-vol -k <private_keyfile> -c <certificate_file> -u <user_id>
```



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



**Note**

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



**Note**

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

## Example

This command bundles an image created in a loopback file:

```
# ec2-bundle-image -i my-image.fs -k pk-HKZYKTAIG2ECMXIYBH3HXV4ZBZQ55CLO.pem
-c cert-HKZYKTAIG2ECMXIYBH3HXV4ZBZQ55CLO.pem -u 495219933132
image.part.00
image.part.01
...
image.part.NN
image.manifest.xml
```

This command bundles the local machine root file system:

```
# ec2-bundle-vol -k pk-HKZYKTAIG2ECMXIYBH3HXV4ZBZQ55CLO.pem -c cert-
HKZYKTAIG2ECMXIYBH3HXV4ZBZQ55CLO.pem -u 495219933132
image.part.00
image.part.01
...
image.part.NN
image.manifest.xml
```

## Uploading a Bundled AMI

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

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

### To upload the bundled AMI

- Enter the following command:

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

where `<bucket>` is the target bucket, `<access_key>` is your AWS Access Key, and `<secret_key>` is your AWS Secret Key.

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

## Sharing AMIs

### Topics

- [Shared AMI Guidelines](#)
- [Sharing AMIs](#)
- [Making an AMI Public](#)
- [Sharing an AMI with Specific Users](#)
- [Publishing Shared AMIs](#)

This section describes how to build and share AMIs.

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

For information on building shared AMIs, see [Shared AMI Guidelines](#). For information on sharing AMIs, see [Sharing AMIs](#).

## Shared AMI Guidelines

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

To build a shared AMI, follow these guidelines:

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



### Note

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

## Update the AMI Tools at Boot Time

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

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

- Add the following to `rc.local`:

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

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



#### Note

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



#### Note

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

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

## Disable Password-Based Logins for Root

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

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

### To disable password-based logins for root

1. Open the `/etc/ssh/sshd_config` file with a text editor and change the following line:

```
#PermitRootLogin yes
```

to

```
PermitRootLogin without-password
```

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

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

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



#### Note

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



#### Note

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

## Install Public Key Credentials

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

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

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

```
if [ ! -d /root/.ssh ] ; then
    mkdir -p /root/.ssh
    chmod 700 /root/.ssh
fi
# Fetch public key using HTTP
curl http://169.254.169.254/2007-08-29//meta-data/public-keys/0/openssh-key >
/tmp/my-key
if [ $? -eq 0 ] ; then
    cat /tmp/my-key >> /root/.ssh/authorized_keys
    chmod 600 /root/.ssh/authorized_keys
    rm /tmp/my-key
fi
```

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



#### Note

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

## Disable sshd DNS Checks (optional)

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

### To disable sshd DNS checks

- Open the `/etc/ssh/sshd_config` file with a text editor and change the following line:

```
#UseDNS yes
```

to:

```
UseDNS no
```



### Note

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

## Identify Yourself

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

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

## Protect Yourself

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

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

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



### Note

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

## Sharing AMIs

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



### Note

Before proceeding, make sure to read the security considerations of sharing AMIs in the [Shared AMI Guidelines](#) section.

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

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



### Note

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

Select from the following:

1. [Making an AMI Public](#)
2. [Sharing an AMI with Specific Users](#)
3. [Publishing Shared AMIs](#)

## Making an AMI Public

### To make an AMI public

- Add the `all` group to the AMI's `launchPermission` attribute using the [ec2-modify-image-attribute](#) command:

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

where `<ami_id>` is the ID of the AMI.

### To check the launch permissions of an AMI

- Use the [ec2-describe-image-attribute](#) command:

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

where `<ami_id>` is the ID of the AMI.

### To make an AMI private again

- Remove the `all` group from its launch permissions:

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

where `<ami_id>` is the ID of the AMI.

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

### Example

This example makes the `ami-2bb65342` AMI public:

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

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

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

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

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

## Sharing an AMI with Specific Users

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

### To grant explicit launch permissions

- Enter the following command:

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

where `<ami_id>` is the ID of the AMI and `<user_id>` is the user's account ID, without hyphens.

### To remove launch permissions for a user

- Enter the following command:

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

where `<ami_id>` is the ID of the AMI and `<user_id>` is the user's account ID, without hyphens.

### To remove all launch permissions

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

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

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



#### Note

The AMI owner always has rights to the AMI and will be unaffected by the [ec2-reset-image-attribute](#) command.

### Example

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

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

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

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

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

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

## Publishing Shared AMIs

After creating a shared AMI, other developers can find it in the [EC2 Resource Center](#).

To publish your AMI, post it in the Public AMIs Folder of the [Amazon Web Services Resource Center](#).

You must include the following information when publishing AMIs:

- AMI ID
- AMI manifest

We recommend including the following information when publishing AMIs:

- Publisher
- Publisher URL
- OS / Distribution
- Key Features
- Description



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

## Creating Paid AMIs

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

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



#### Basic DevPay Flow

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

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

## Paid and Supported AMIs

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

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

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

## Summary of How Paid AMIs Work

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




#### Note

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

#### Paid AMI Process

## Amazon Elastic Compute Cloud Developer Guide

### Summary of How Supported AMIs Work

1	You create an AMI as described elsewhere in this guide.
2	You <a href="#">register a product</a> with Amazon DevPay. As part of this process, you provide a product description, product pricing, etc. This registration process creates a product code for the product and a URL where customers can sign up to use the product (called the <i>purchase URL</i> ).
3	You use an Amazon EC2 command or API call to <a href="#">associate the product code</a> with your AMI. This makes the AMI a paid AMI.
4	<p>You use an Amazon EC2 command or API call to <a href="#">share the AMI</a> with select customers or the public.</p> <div><b>Note</b></div> <p>Even if you share a paid AMI and it has a product code, no one can use the AMI until they sign up for it (see the following steps).</p>
5	You advertise your paid AMI to customers and make the purchase URL available to them. Note that you can submit your paid AMI to be listed on the AWS Resource Center with other public AMIs at <a href="http://developer.amazonwebservices.com/connect/kbcategory.jspa?categoryID=116">http://developer.amazonwebservices.com/connect/kbcategory.jspa?categoryID=116</a> .
6	Customers who want to use your AMI discover your product through your advertisements or the AWS Resource Center, etc.
7	Customers then use the purchase URL you provide to sign up for and purchase your product. If they're not already signed up for Amazon EC2, they'll be prompted to sign up. They purchase your product with their Amazon.com accounts. They must have the credentials needed to launch Amazon EC2 instances. At this point, they have the AMI ID (from either step 5 or step 6 above).
8	Customers then launch an Amazon EC2 instance specifying the AMI ID. Because you associated the shared AMI with the product code, the customers are charged at the rate you set. For more information, see <a href="#">Paying for AMIs</a>

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

## Summary of How Supported AMIs Work

The following steps summarize the basic process for supported AMIs:

### Supported AMI Process

1	You <a href="#">register a product</a> with Amazon DevPay. As part of this process, you provide a product description, product pricing, etc. This registration process creates a product code for the product and a URL where customers can sign up to use the product (called the <i>purchase URL</i> ).
2	You then advertise your paid AMI to customers and make the purchase URL available to them.
3	Customers who want to use your product discover the product through your advertisements, etc.

4	Customer then use the purchase URL to sign up for and purchase your product. If they're not already signed up for Amazon EC2, they'll be prompted to sign up. They purchase your product with their Amazon.com accounts. They must have the credentials needed to launch Amazon EC2 instances. At this point, they have the product code (from either step 2 or step 3 above).
5	Customers then use an Amazon EC2 command or API call to <a href="#">associate the product code</a> with their AMIs.
6	Customers then launch one or more instances of the AMIs. Because the customers associated their AMIs with the product code, they are charged at the rate you set.

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

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

## Product Registration

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



### Note

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

You provide the following information during registration:

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

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

Registration provides you with the following information:

- Product code
- Product token
- Purchase URL

You need the product code and purchase URL to integrate your product with DevPay as described in [Summary of How Paid AMIs Work](#) and [Summary of How Supported AMIs Work](#). You need the

product token if you're going to set up your system to later verify whether a customer is still subscribed to your product. For more information, see the *Amazon DevPay Developer Guide*.



### Important

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

## Associating a Product Code with an AMI

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

### To associate a product code with an AMI

- Enter the following command:

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

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

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

- Enter the following command:

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

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

### Example

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

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

productCodes	ami-5bae4b32	productCode	774F4FF8
--------------	--------------	-------------	----------

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

```
PROMPT> ec2-describe-image-attribute ami-5bae4b32 --product-code
```

productCodes	ami-5bae4b32	productCode	774F4FF8
--------------	--------------	-------------	----------

## Sharing Your Paid AMI with Select Users or the Public

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

### To share the AMI

- Enter the following command:

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

where `<ami_id>` is the AMI ID.

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

### Example

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

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

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

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



### Note

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

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

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

- Enter the following command:

```
PROMPT> ec2-confirm-product-instance <product_code> -i <instance>
```

where `<product_code>` is the product code and `<instance>` is the instance.

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

### Example

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

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

## Getting the Product Code from Within an Instance

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

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

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

Example response:

```
774F4FF8
```

# Launching and Using Instances

## Topics

- [Instance Usage](#)
- [Instance Types](#)
- [Instance Metadata](#)
- [Using Shared AMIs](#)
- [Paying for AMIs](#)
- [Get Console Output and Reboot Instances](#)

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

## Instance Usage

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

For information about available instance types, see [Selecting Instance Types](#).

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

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

- Do not rely on an instance's local storage for valuable, long-term data.  
When instances fail, the data on the local disk is lost. Use a replication strategy across multiple instances to keep your data safe or store your persistent data in Amazon S3
- Define images based on the type of work they perform.  
For "internet applications," you might define one image for database instances and another for web servers. Image creation and storage are cheap and easy operations, so you can individualize and customize as necessary. Specialized images can result in smaller AMI sizes, which will boot considerably faster.



- Monitor the health of your instances.  
You can make your instances work for you by configuring them to monitor each other. For example, you could create an image that contains an open-source monitoring tool such as Nagios or OpenNMS. Then, your other instances could report their health to the monitoring instance.
- Keep your Amazon EC2 firewall permissions as restrictive as possible.  
Only open up permissions that you require. Use separate groups to deal with instances that have different security requirements. Consider using additional security measures inside your instance (such as using your own firewall). If you need to log in interactively (ssh), consider creating a bastion security group that allows external login and keep the remainder of your instances in a group that does not allow external login.

## Instance Types

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

## Available Instance Types

The following instance types are available:

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



### Note

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

## Measuring Compute Resources

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

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



### Note

We use several internal benchmarks and tests to manage the consistency and predictability of the performance of an Amazon EC2 Compute Unit. For more information, refer to the [Instance page](#).

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

## I/O Resources

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

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

## Instance Metadata

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

### Example

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

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

## Categories of Available Data

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

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

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

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



### Note

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

## Data Retrieval

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



### Note

Amazon EC2 Version 1.0 is part of a legacy versioning scheme. Newer versions follow a date based versioning scheme. For more information on the versioning scheme used by Amazon EC2, see [API Versioning](#).

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

## Security of Launch Data

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



### Note

You are not billed for HTTP requests used to retrieve metadata and user-supplied data.

## Retrieving Metadata

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

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

### Example

The following examples list HTTP GET requests and responses. You can use a tool such as `curl` or `wget` to make these types of requests.

This example gets the available API versions:

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

This example gets the top-level metadata items:

```
GET http://169.254.169.254/2007-08-29/meta-data/
ami-id
ami-launch-index
ami-manifest-path
hostname
instance-id
instance-type
local-hostname
```

```
local-ipv4  
public-hostname  
public-ipv4  
public-keys/  
reservation-id  
security-groups
```

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

```
GET http://169.254.169.254/2007-08-29/meta-data/ami-manifest-path  
my-amis/spamd-image.manifest.xml  
GET http://169.254.169.254/2007-08-29/meta-data/ami-manifest-path  
my-amis/spamd-image.manifest.xml  
GET http://169.254.169.254/2007-08-29/meta-data/ami-id  
ami-2bb65342  
GET http://169.254.169.254/2007-08-29/meta-data/reservation-id  
r-fea54097  
GET http://169.254.169.254/2007-08-29/meta-data/hostname  
domU-12-34-31-00-00-05.usma1.compute.amazonaws.com
```

This example gets the list of available public keys:

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

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

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

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

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

This example gets the product code:

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

## Retrieving User Data

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



### Note

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

### Example

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

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

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

```
GET http://169.254.169.254/2007-08-29/user-data

[general]
instances: 4

[instance-0]
s3-bucket: fred

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

## Example: AMI Launch Index Value

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

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

```
store-size=123PB backup-every=5min | replicate-every=1min | replicate-
every=2min | replicate-every=10min | replicate-every=20min
```

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

Alice launches four instances:

```
$ ec2-run-instances ami-2bb65342 -n 4 -d "store-size=123PB backup-every=5min
| replicate-every=1min | replicate-every=2min | replicate-every=10min | rep
licate-every=20min"

RESERVATION      r-fea54097          598916040194      default
INSTANCE i-3ea74257 ami-2bb65342 pending 0 m1.small 2007-08-07T11:29:58+0000
INSTANCE i-31a74258 ami-2bb65342 pending 1 m1.small 2007-08-07T11:29:58+0000
INSTANCE i-31a74259 ami-2bb65342 pending 2 m1.small 2007-08-07T11:29:58+0000
INSTANCE i-31a7425a ami-2bb65342 pending 3 m1.small 2007-08-07T11:29:58+0000
```

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

- AMI id: ami-2bb65342
- AMI manifest path: ec2-public-images/getting-started.manifest.xml
- Reservation ID: r-fea54097
- Public keys: none
- Security group names: default
- Instance type: m1.small

However each instance has certain unique metadata:

*Instance 1*

## Amazon Elastic Compute Cloud Developer Guide

### Example: AMI Launch Index Value

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

#### Instance 2

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

#### Instance 3

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

#### Instance 4

Metadata	Value
instance-id	i-31a7425a
ami-launch-index	3
hostname	domU-12-31-33-00-01-74.usma1.compute.amazonaws.com
local-ipv4	216.182.228.90

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

#### Metadata Discovery Process

1	Determine which instance in the launch group it is:  GET <code>http://169.254.169.254/2007-08-29/meta-data/ami-launch-index</code> 1
---	---

2	Retrieve the user data:  GET http://169.254.169.254/2007-08-29/user-data store-size=123PB backup-every=5min   replicate-every=1min   replicate-every=2min   replicate-every=10min   replicate- every=20min
3	Extract the appropriate part of the user data:  user_data.split(' ')[ami_launch_index]

## Instance Storage

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

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

We highly recommend backing up important data to Amazon S3.

## Storage Locations

Storage is exposed on the following instance types as follows:

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

## Disk Performance Optimization

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

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

```
dd if=/dev/zero of=/dev/sdb bs=1M
dd if=/dev/zero of=/dev/sdc bs=1M
dd if=/dev/zero of=/dev/sdd bs=1M (m1.xlarge only)
dd if=/dev/zero of=/dev/sde bs=1M (m1.xlarge only)
```



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

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



**Note**

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

## RAID Configuration

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

```
echo $((30*1024)) > /proc/sys/dev/raid/speed_limit_min
```



**Note**

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

# Using Shared AMIs

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

## Finding Shared AMIs

### To find shared AMIs

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

### Example

This command displays a list of all public AMIs.

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

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

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

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

AMIs that you own are excluded from the list.

This command displays a list of AMIs owned by Amazon:

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

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

```
PROMPT> ec2dim -o <target_uid>
```

where `<target_uid>` is the account ID of the user who owns the AMIs for which you are looking.

## Safe Use of Shared AMIs

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

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

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



#### Note

Users cannot alias an AMI's owner.

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

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

## Paying for AMIs

- [Finding Paid AMIs](#)
- [Purchasing a Paid AMI](#)
- [Launching Paid AMIs](#)
- [Paying for Support](#)
- [Bills for Paid and Supported AMIs](#)

Amazon EC2 integrates with Amazon DevPay, allowing developers to charge users for the use of their AMIs or to provide support for instances. To learn more about Amazon DevPay see the *Amazon DevPay Developer Guide*. To learn more about charging for your use of your AMIs, or providing support, see [Creating Paid AMIs](#)

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

## Finding Paid AMIs

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

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



#### Note

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

### To check if an AMI is paid:

- Enter this command:

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

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

The command returns the following:

```
IMAGE <ami_id> <manifest> <user_id>, <status> {private | public}  
<product_code>
```

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

### Example

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

```
$ ec2-describe-images ami-2bb65342  
IMAGE ami-2bb65342 awesome-ami/webserver.manifest.xml 495219933132 available  
private 774F4FF8
```

## Purchasing a Paid AMI

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

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

## Launching Paid AMIs

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

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

### To launch a paid AMI

- Enter the following command:

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

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



#### Note

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

#### Example

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

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

## Paying for Support

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

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

- Use the `ec2-modify-image-attribute` command:

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

where `<ami_id>` is the AMI ID and `<product_code>` is the product code.



#### Important

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

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

#### Example

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

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

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

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

## Bills for Paid and Supported AMIs

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

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

## Get Console Output and Reboot Instances

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

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

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

## Get Console Output

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

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



### Note

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

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



### Note

Only the instance owner can access the console output.

## Reboot Instances

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

# Instance Addressing and Network Security

## Topics

- [Instance Addressing](#)
- [Network Security](#)

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

## Instance Addressing

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

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

For more information, see [Instance Addressing](#).



### Note

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

## Private (RFC 1918) Addresses

All Amazon EC2 instances are allocated a private address by DHCP. These addresses are from a range

defined in "[RFC 1918 - Address Allocation for Private Internets](#)", are only routable within Amazon EC2, and are used for communication between instances.

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

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

### To determine your IP address

1. Connect to the instance.
2. Enter one of the following:
  - ```
PROMPT> ifconfig eth0
```
  - ```
PROMPT> curl http://169.254.169.254/latest/meta-data/local-ipv4
```

The second option refers to the instance data (see [Instance Metadata](#)).

## Internal DNS Name

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

## Public Addresses

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

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

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



### Important

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

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

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



## Public DNS

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



### Important

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

## Network Security

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

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

## Concepts

### Security Groups

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

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



### Note

You can create up to 100 rules per group.

### Group Membership

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

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

### Group Access Rights

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

## Examples

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



### Note

These examples use the [command line tools](#). You can achieve the same results using the [SOAP API](#).

## Default Group

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

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

```
$ ec2-run-instances ami-eca54085
INSTANCE i-cfd732a6 ami-eca54085 pending 0 m1.small
2007-07-11T16:40:44+0000
```

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
598916040194  GRPNAME default
```

	FROM	USER
--	------	------

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

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

```
$ nmap -P0 -p1-100 domU-12-31-33-00-01-56.usma1.compute.amazonaws.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. Albert repeats 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 about the security of her subscriber database, so she wants to restrict network access to her middle and back tier machines. When the traffic to her site increases over the holiday shopping period, she adds additional instances to her web and application tiers to handle the extra load.

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

```
$ ec2-add-group apache -d "Mary's Apache group"
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
```

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

```
$ ec2run ami-fba54092 -n 7 -g apache
INSTANCE i-cfd732a6 ami-fba54092 pending 0 m1.small
2007-07-11T16:40:44+0000
INSTANCE i-cfd732a7 ami-fba54092 pending 0 m1.small
2007-07-11T16:40:44+0000
INSTANCE i-cfd732a8 ami-fba54092 pending 0 m1.small
2007-07-11T16:40:44+0000
INSTANCE i-cfd732a9 ami-fba54092 pending 0 m1.small
2007-07-11T16:40:44+0000
INSTANCE i-cfd732aa ami-fba54092 pending 0 m1.small
2007-07-11T16:40:44+0000
INSTANCE i-cfd732ab ami-fba54092 pending 0 m1.small
2007-07-11T16:40:44+0000
INSTANCE i-cfd732ac ami-fba54092 pending 0 m1.small
2007-07-11T16:40:44+0000

$ ec2din i-cfd732a6
RESERVATION    r-0592776c    598916040194
INSTANCE    i-cfd732a6    ami-fba54092    domU-
12-31-33-00-04-16.usma1.compute.amazonaws.com    running 0
m1.small 2007-07-11T16:40:44+0000
```

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

```
$ nmap -P0 -pl-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
```

4. Mary verifies her web server can be reached:

```
$ 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 '^]'.
```

Mary can reach her web server.

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

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

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

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

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

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

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

```
$ nmap -P0 -p1-100 domU-12-31-33-00-03-D1.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
```

9. Mary confirms that her web servers have access to her application servers:

- a. She (temporarily) grants SSH access from her workstation to the web server group:

```
$ 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
```

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

## Notes

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

# Using the APIs

## Topics

- [Using the SOAP API](#)
- [Using the Query API](#)

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

## Using the SOAP API

### Topics

- [WSDL and Schema Definitions](#)
- [Programming Language Support in Amazon EC2](#)
- [Request Authentication](#)
- [The Response Structure](#)
- [Web Services References](#)

## WSDL and Schema Definitions

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

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



### Note

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

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

## Programming Language Support in Amazon EC2

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

## Request Authentication

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

The following is the content of an insecure RunInstances operation:

```
<RunInstances xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
  <instancesSet>
    <item>
      <imageId>ami-60a54009</imageId>
      <minCount>1</minCount>
      <maxCount>3</maxCount>
    </item>
  </instancesSet>
  <groupSet/>
</RunInstances>
```

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

The secure version of the request begins with the following:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <wsse:BinarySecurityToken
        xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
        Encoding="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-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>HhRbxBBmc200348f8nLNZyo4AOM=</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>

```

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

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

## The Response Structure

In response to a request, the Amazon EC2 web service returns an XML data structure that conforms to an XML schema defined as part of the Amazon EC2 [WSDL](#). The structure of a XML response is specific to the associated request. In general, the response data types are named according to the

operation performed and whether the data type is a container (can have children). Examples of containers include `groupSet` for security groups and `instancesSet` for instances. Item elements are children of containers and their contents vary according to the container's role.

An example response is:

```
<RunInstancesResponse xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
  <reservationId>r-47a5402e</reservationId>
  <ownerId>UY93TLBUXIEON5NQVUUX6OMPWBZIQNFM</ownerId>
  <groupSet>
    <item>
      <groupId>default</groupId>
    </item>
  </groupSet>
  <instancesSet>
    <item>
      <instanceId>i-2ba64342</instanceId>
      <imageId>ami-60a54009</imageId>
      <instanceState>
        <code>0</code>
      </instanceState>
      <name>pending</name>
      <dnsName></dnsName>
    </item>
    <item>
      <instanceId>i-2bc64242</instanceId>
      <imageId>ami-60a54009</imageId>
      <instanceState>
        <code>0</code>
      </instanceState>
      <name>pending</name>
      <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>
      </instanceState>
      <name>pending</name>
      <dnsName>domU-12-34-28-00-00-5C.dc2.compute.amazonaws.com</dnsName>
      <keyName>example-key-name</keyName>
      <instanceType>m1.small</instanceType>
      <launchTime>2007-08-07T11:54:42.000Z</launchTime>
    </item>
  </instancesSet>
</RunInstancesResponse>
```

## Web Services References

- [Web Service Description Language \(WSDL\)](#)
- [WS-Security BinarySecurityToken Profile](#)

## Using the Query API

### Topics

- [Query Parameters](#)
- [Query API Authentication](#)
- [Example Request](#)



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

## Query Parameters

Each Query request must include some common parameters to handle authentication and selection of an action. For more information, see [Common Query Parameters](#).

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

## Query API Authentication

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

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

### Query Authentication

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



#### Note

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

### To calculate the request signature

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

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

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

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

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

```
ActionDescribeImagesAWSAccessKey
Id10QMXFEV71ZS32XQFTR2SignatureVersion1Timestamp2006-12-08T07:48:03ZVersion20
07-01-03
```

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

```
GjH3941IBe6qsgQu+k7FpCJjpnc=
```

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

```
import java.security.SignatureException;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;

public class HmacExample
{
    private static final String HMAC_SHA1_ALGORITHM = "HmacSHA1";

    /**
     * Computes RFC 2104-compliant HMAC signature.
     *
     * @param data
     *    The data to be signed.
     * @param key
     *    The signing key.
     * @return
     *    The base64-encoded RFC 2104-compliant HMAC signature.
     * @throws
     *    java.security.SignatureException when signature generation fails
     */
    public static String calculateRFC2104HMAC(String data, String key)
        throws java.security.SignatureException
    {
        String result;
        try {
            // get an hmac_sha1 key from the raw key bytes
            SecretKeySpec signingKey = new SecretKeySpec(key.getBytes(),
HMAC_SHA1_ALGORITHM);

            // get an hmac_sha1 Mac instance and initialize with the signing
key
            Mac mac = Mac.getInstance(HMAC_SHA1_ALGORITHM);
            mac.init(signingKey);

            // compute the hmac on input data bytes
```

```
byte[] rawHmac = mac.doFinal(data.getBytes());

// base64-encode the hmac
result = Base64.encodeBytes(rawHmac);
}
catch (Exception e) {
    throw new SignatureException("Failed to generate HMAC : " +
e.getMessage());
}
return result;
}
}
```



#### Note

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

## Example Request

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

```
?AWSAccessKeyId=10QMXFEV71ZS32XQFTR2&Action=DescribeImages&SignatureVersion=1
&Timestamp=2006-12-08T07%3A48%3A03Z&Version=2007-01-03&Signature=69DSJs1z%2B0
wWJmdB77%2BLm0N0Trs%3D
```

# API Reference

## Topics

- [API Conventions](#)
- [API Versioning](#)
- [API Error Codes](#)
- [Common Data Types](#)
- [Amazon EC2 SOAP API](#)
- [Amazon EC2 Query API](#)

Amazon EC2 provides two APIs: [SOAP](#) and [Query](#), which allow you to launch and control instances from your own applications.

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



### Note

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

Before using the API, we recommend that you familiarize yourself with their [conventions](#).

## API Conventions

### Overview

This section describes Amazon EC2 API conventions.

### Actions

Actions encapsulate the possible interactions with Amazon EC2. These can be viewed as remote procedure calls and consist of a request and response message pair. Requests must be signed, allowing

Amazon EC2 to [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:  
<http://ec2.amazonaws.com/doc/2007-08-29/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 Versioning

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

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

SOAP clients that retrieve the Amazon EC2 WSDL at runtime and generate their requests dynamically using that WSDL should reference the WSDL for the version of the API that the client was developed

against. This ensures that the client software continues to work even if backwards incompatible API changes are introduced. The WSDL for each supported API version is available from the following URI:

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

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

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



#### Note

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

## API Error Codes

### Overview

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

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

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

### Summary of Client Error Codes

Error Code	Description	Notes
<code>AuthFailure</code>	User not authorized.	You might be trying to run an AMI for which you do not have permission.
<code>InvalidManifest</code>	Specified AMI has an unparseable Manifest.	
<code>InvalidAMIID.Malformed</code>	Specified AMI ID is not valid.	
<code>InvalidAMIID.NotFound</code>	Specified AMI ID does not exist.	
<code>InvalidAMIID.Unavailable</code>	Specified AMI ID has been deregistered and is no longer available.	
<code>InvalidInstanceID.Malformed</code>	Specified instance ID is not valid.	
<code>InvalidInstanceID.NotFound</code>	Specified instance ID does	

**Amazon Elastic Compute Cloud Developer Guide**  
**Summary of Client Error Codes**

Error Code	Description	Notes
	not exist.	
<code>InvalidKeyPair.NotFound</code>	Specified key pair name does not exist.	
<code>InvalidKeyPair.Duplicate</code>	Attempt to create a duplicate key pair.	
<code>InvalidGroup.NotFound</code>	Specified group name does not exist.	
<code>InvalidGroup.Duplicate</code>	Attempt to create a duplicate group.	
<code>InvalidGroup.InUse</code>	Specified group can not be deleted because it is in use.	
<code>InvalidGroup.Reserved</code>	Specified group name is a reserved name.	
<code>InvalidParameterValue</code>	The value supplied for a parameter was invalid.	Requests that could cause this error include (for example) supplying an invalid image attribute to the <code>DescribeImageAttribute</code> request or an invalid version or encoding value for the <code>userData</code> in a <code>RunInstances</code> request.
<code>InvalidPermission.Duplicate</code>	Attempt to authorize a permission that has already been authorized.	
<code>InvalidPermission.Malformed</code>	Specified permission is invalid.	
<code>InvalidReservationID.Malformed</code>	Specified reservation ID is invalid.	
<code>InvalidReservationID.NotFound</code>	Specified reservation ID does not exist.	
<code>InstanceLimitExceeded</code>	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.
<code>InvalidParameterCombination</code>	<code>RunInstances</code> was called with <code>minCount</code> and <code>maxCount</code> set to 0 or <code>minCount &gt; maxCount</code> .	
<code>InvalidUserID.Malformed</code>	The user ID is neither in the form of an AWS account ID or one of the special values accepted by the owner or	

Error Code	Description	Notes
	<code>executableBy</code> flags in the <code>DescribeImages</code> call.	
<code>InvalidAMIAttributeItemValue</code>	The value of an item added to, or removed from, an image attribute is invalid.	If you are specifying a <code>userId</code> , check that it is in the form of an AWS account ID.
<code>UnknownParameter</code>	An unknown or unrecognized parameter was supplied.	Requests that could cause this error include supplying a misspelled parameter or a parameter that is not supported for the specified API version.

## Summary of Server Error Codes

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

## Common Data Types

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

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

## DescribeImagesResponseItemType

The `DescribeImagesResponseItemType` data type.

## Relevant Operations

Operations that use this data type include:



- [DescribeImages](#)

## Contents

The following table describes the elements contained in DescribeImagesResponseItemType.

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

## DescribeKeyPairsResponseItemType

The DescribeKeyPairsResponseItemType data type.

### Relevant Operations

Operations that use this data type include:

- [DeleteKeypair](#)
- [DescribeKeypairs](#)

## Contents

The following table describes the elements contained in DescribeKeyPairsResponseItemType.

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

Name	Description
	the private key. Type: xsd:string

## EmptyElementType

The EmptyElementType data type.

### Relevant Operations

Operations that use this data type include:

- [ResetImageAttribute](#)
- [DescribeImageAttribute](#)

### Contents

The empty element has no contents.

## GroupSetType

The GroupSetType data type.

### Relevant Operations

Operations that use this data type include:

- [RunInstances](#)

### Contents

The following table describes the elements contained in GroupSetType.

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

## InstanceStateType

The InstanceStateType data type.

### Relevant Operations

Operations that use this data type include:

- [RunInstances](#)

- [DescribeInstances](#)
- [TerminateInstances](#)

## Contents

The following table describes the elements contained in InstanceStateType.

Name	Description
code	<p>A 16-bit unsigned integer. The high byte is an opaque internal value and should be ignored. The low byte is set based on the state represented:</p> <ul style="list-style-type: none"><li>• 0: pending</li><li>• 16: running</li><li>• 32: shutting-down</li><li>• 48: terminated</li></ul> <p>Type: xsd:int</p>
state	<p>The current state of the instance.</p> <ul style="list-style-type: none"><li>• <code>pending</code>: the instance is in the process of being launched</li><li>• <code>running</code>: the instance launched (although the boot process might not be completed)</li><li>• <code>shutting-down</code>: the instance started shutting down</li><li>• <code>terminated</code>: the instance terminated</li></ul> <p>Type: xsd:string</p>

## IpPermissionType

The IpPermissionType data type.

## Relevant Operations

Operations that use this data type include:

- [AuthorizeSecurityGroupIngress](#)
- [DescribeSecurityGroups](#)
- [RevokeSecurityGroupIngress](#)

## Contents

The following table describes the elements contained in IpPermissionType.

Name	Description
ipProtocol	<p>IP protocol.</p> <p>Type: xsd:string</p>

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

## LaunchPermissionItemType

The LaunchPermissionItemType data type.

### Relevant Operations

Operations that use this data type include:

- [ModifyImageAttribute](#)
- [DescribeImageAttribute](#)

### Contents

The following table describes the elements contained in LaunchPermissionItemType.

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

## LaunchPermissionOperationType

The LaunchPermissionOperationType data type.

## Relevant Operations

Operations that use this data type include:

- [ModifyImageAttribute](#)

## Contents

The following table describes the elements contained in LaunchPermissionOperationType.

Name	Description	Required
add	Adds launch permissions. Type: <a href="#">ec2:LaunchPermissionItemType</a> []	Choice between add and remove
remove	Removes launch permissions. Type: <a href="#">ec2:LaunchPermissionItemType</a> []	Choice between add and remove

## ProductCodeItemType

The ProductCodeItemType data type.

## Relevant Operations

Operations that use this data type include:

- [ModifyImageAttribute](#)
- [DescribeImageAttribute](#)

## Contents

The following table describes the elements contained in ProductCodeItemType.

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

## ProductInstanceResponseItemType

The ProductInstanceResponseItemType data type.

## Relevant Operations

Operations that use this data type include:

- [ConfirmProductInstance](#)

## Contents

The following table describes the elements contained in ProductInstanceResponseItemType.

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

## ReservationInfoType

The ReservationInfoType data type.

## Relevant Operations

Operations that use this data type include:

- [RunInstances](#)
- [DescribeInstances](#)

## Contents

The following table describes the elements contained in ReservationInfoType.

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

## RunInstanceItemType

The RunInstanceItemType data type.

## Relevant Operations

Operations that use this data type include:

- [RunInstances](#)

## Contents

The following table describes the elements contained in RunInstanceItemType.

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

## RunningInstancesItemType

The RunningInstancesItemType data type.

## Relevant Operations

Operations that use this data type include:

- [RunInstances](#)

## Contents

The following table describes the elements contained in RunningInstancesItemType.

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

Name	Description
instanceState	<p>The current state of the instance.</p> <ul style="list-style-type: none"> <li>• <code>pending</code>: the instance is in the process of launching</li> <li>• <code>running</code>: the instance launched (although it the boot process might not be complete)</li> <li>• <code>shutting-down</code>: the instance is shutting down</li> <li>• <code>terminated</code>: the instance terminated</li> </ul> <p>Type: <a href="#">ec2:InstanceStateType</a></p>
privateDnsName	<p>The private DNS name assigned to the instance. This DNS name can only be used inside the Amazon EC2 network. This element remains empty until the instance enters a running state. For more information, see <a href="#">Instance Addressing and Network Security</a>.</p> <p>Type: <code>xsd:string</code></p>
dnsName	<p>The public DNS name assigned to the instance. This DNS name is contactable from outside the Amazon EC2 network. This element remains empty until the instance enters a running state. For more information, see <a href="#">Instance Addressing and Network Security</a>.</p> <p>Type: <code>xsd:string</code></p>
reason	<p>Optional. Reason for the most recent state transition. This might be an empty string.</p> <p>Type: <code>xsd:string</code></p>
keyName	<p>Optional. If this instance was launched with an associated key pair, this displays the key pair name.</p> <p>Type: <code>xsd:string</code></p>
amiLaunchIndex	<p>Optional. The AMI launch index, which can be used to find this instance within the launch group. For more information, see <a href="#">Instance Metadata</a>.</p> <p>Type: <code>xsd:string</code></p>
productCodes	<p>Optional. Product codes attached to this instance.</p> <p>Type: <a href="#">ec2:ProductCodeItemType</a>[]</p>
instanceType	<p>The instance type. For more information on instance types, see <a href="#">Selecting Instance Types</a></p> <p>Type: <code>xsd:string</code></p>
launchTime	<p>The time the instance launched.</p> <p>Type: <code>xs:dateTime</code></p>

## SecurityGroupItem



The SecurityGroupItemType data type.

## Relevant Operations

Operations that use this data type include:

- [DescribeSecurityGroups](#)

## Contents

The following table describes the elements contained in SecurityGroupItemType.

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

## TerminateInstancesResponseInfoType

The TerminateInstancesResponseInfoType data type.

## Relevant Operations

Operations that use this data type include:

- [TerminateInstances](#)

## Contents

The following table describes the elements contained in TerminateInstancesResponseInfoType.

Name	Description
instanceId	Instance ID returned from previous call to <a href="#">RunInstances</a> . Type: xsd:string

## UserDataType

The UserDataType data type.

## Relevant Operations

Operations that use this data type include:

- [RunInstances](#)

## Contents

The following table describes the elements contained in UserDataType.

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

## Notes

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

## UserIdGroupPairType

The UserIdGroupPairType data type.

## Relevant Operations

Operations that use this data type include:

- [AuthorizeSecurityGroupIngress](#)
- [DescribeSecurityGroups](#)
- [RevokeSecurityGroupIngress](#)

## Contents

The following table describes the elements contained in UserIdGroupPairType.

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

Name	Description
	Type: xsd:string

## Amazon EC2 SOAP API

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

### List of Operations by Function

#### Images

- [RegisterImage](#)
- [DescribeImages](#)
- [DeregisterImage](#)

#### Instances

- [RunInstances](#)
- [DescribeInstances](#)
- [TerminateInstances](#)
- [ConfirmProductInstance](#)

#### Key Pairs

- [CreateKeyPair](#)
- [DescribeKeyPairs](#)
- [DeleteKeyPair](#)

#### Image Attributes

- [ModifyImageAttribute](#)
- [DescribeImageAttribute](#)
- [ResetImageAttribute](#)

#### Security Groups

- [CreateSecurityGroup](#)
- [DescribeSecurityGroups](#)
- [DeleteSecurityGroup](#)
- [AuthorizeSecurityGroupIngress](#)
- [RevokeSecurityGroupIngress](#)

## AuthorizeSecurityGroupIngress

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

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

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

### Request Parameters

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

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

### Response Elements

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

Name	Description
<code>return</code>	true if permissions successfully added. Type: <code>xsd:boolean</code>

### Sample Request

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

```
</item>
</ipPermissions>
</AuthorizeSecurityGroupIngress>
```

## Sample Response

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

## Related Operations

- [CreateSecurityGroup](#)
- [DescribeSecurityGroups](#)
- [RevokeSecurityGroupIngress](#)
- [DeleteSecurityGroup](#)

## ConfirmProductInstance

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

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

## Request Parameters

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

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

## Response Elements

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

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

Name	Description
	is attached to the instance. Type: xsd:string

## Sample Request

```
<ConfirmProductInstance xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
  <productCode>774F4FF8</productCode>
  <instanceId>i-10a64379</instanceId>
</ConfirmProductInstance>
```

## Sample Response

```
<ConfirmProductInstanceResponse xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
  <return>true</return>
  <ownerId>254933287430</ownerId>
</ConfirmProductInstanceResponse>
```

## Related Operations

- [DescribeInstances](#)
- [RunInstances](#)

## CreateKeyPair

The `CreateKeyPair` operation creates a new 2048 bit RSA key pair and returns a unique ID that can be used to reference this key pair when [launching](#) new instances.

## Request Parameters

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

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

## Response Elements

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

Name	Description
keyName	The key pair name provided in the original request. Type: xsd:string

Name	Description
keyFingerprint	A SHA-1 digest of the DER encoded private key. Type: xsd:string
keyMaterial	An unencrypted PEM encoded RSA private key. Type: xsd:string

## Sample Request

```
<CreateKeyPair xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
  <keyName>example-key-name</keyName>
</CreateKeyPair>
```

## Sample Response

```
<CreateKeyPairResponse xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
  <keyName>example-key-name</keyName>
  <keyFingerprint>1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:f1:6f</keyFingerprint>
  <keyMaterial>-----BEGIN RSA PRIVATE KEY-----
MIIEOQIBAAKCAQBULFg5ujHrtmljnutSuo08Xe56LlT+HM8v/xkaa39EstM3/aFxTHgElQiJLChp
HungXQ29VtC8rc1bW0lkdi23OH5eqkMHGhvEwga0HWASUMl14o3o/IX+0f2UcPoKCOVUR+jx7lSg
5AU52EQfanIn3ZQ8lFW7Edp5a3q4DhJGLUKToHVbicL5E+g45zfB95wIyywWZfEW/UUF3LpGZyq/
ebIUlq1qTbHkLbCC2r7RTn8vpQWp47BGVYgtGSBMpTRP5hnbzZuqj3itkiLHjU39S2sJCJ0TrJx5
i8BygR4s3mHKBj8l+ePQxG1kGbF6R4yg6sECmXn17MRQVXODNHZbAgMBAAECggEAYltsiUsIwDl5
9lCXlRkYGuVfLyLflXenxfI50mDFms/mumTqloHO7tr0oriHDR5K7wMcY/YY5YkcXNo7mvUVDlpM
ZNUJ7r7w9gZRT7f7LylaJ58kOcyajw8TsC4e4LPbFaHwSld6K8rXh64o6WgW4SrsB6ICmr1kGQI7
3wcfgt5ecIu4Tzf00E9IHjn+2eRlSrjBdeORi7KiUNC/pAG23I6MdDOFEQRcCSigCj+4/mciFUSA
SWS4dMbrpb9FNSIcf9dcLxVM7/6KxgJNfZc9XWzUw77Jg8x92Zd0fVhHoux5IzC+UvSKWB4dyfcI
tE8C3p9bbU9VGyY5vLCAiIb4qQKBgQDLiO24GXrIkswF32YtBBMuVgLGcWU9h9HlO9mKAc2m8Cm1
jUE5IpzRjTcd9I2qiIMUTwtgnw42auSCzbUeYmURPtDqyQ7p6AjMuJp9EPemcSVOK9vXYL0Ptco
xW9MC0dtV6iPkCN7gOqiZXPRKaFbWADp16p8UAIvS/a5XXk5jwKBgQCKkphi2EiShluRkhxljyWC
iDCiK6JBRsMvpLbc0v5dKwP5alo1fmdR5PJaV2qvZSj5CYNpMay1/EDNTY5OSIJU+0KfMqbyhsbm
rdLNLDL4+Tcnt7c62/aH01ohYaf/VcBrtLlBfgQoQc7+sAc8vmKkesnF7CqCEKdyF/dhrxYdQKB
gC0iZzzNAapayz1+JcVTwwEid6j9JqNXbBc+Z2YwMi+T0Fv/P/hwkX/ypeOXnIUcw0Ih/YtGBVAC
DQbsz7LcY1HqXiHKYNWNvXgwwO+oiChjxvEkSdsTTIfnK4VSCvU9BxDbQHjdiNDJbL6oar92UN7V
rBYvChJZF7LvUH4YmVpHAoGAb22X7XvoeEO+uZ58/BGKOIGHByHBDiXtzmhdJr15HTYjxK7OgTZm
gK+8zp4L9IbvLGDMJO8vft32XPEWuvI8twCzFH+CSWLQADZMKSSBasOZ/h1FwhdMgCMcY+Qlzd4
JZKjTSu3i7vhvx6RzdSedXEMNTZWN4qlIx3kR5aHcukCgYA9T+ZrvmlF0seQPbLknn7EqhXIjBaT
P8TTvW/6bdPi23ExzxZn7K0drfclYRph1LHMPaONv/x2xALIf91UB+v5ohyloDoasL0gi1jhouRe
2ERKKdwz0ZL9Swq6VTdhr/5G994CK72fy5WhyERbDJUIdHaK3M849Jjuf8cSrvSb4g==
-----END RSA PRIVATE KEY-----</keyMaterial>
</CreateKeyPairResponse>
```

## Related Operations

- [DescribeKeyPairs](#)
- [DeleteKeyPair](#)
- [RunInstances](#)

## CreateSecurityGroup

The CreateSecurityGroup operation creates a new security group.

Every instance is launched in a security group. If no security group is specified during launch, the instances are launched in the default security group. Instances within the same security group have unrestricted network access to each other. Instances will reject network access attempts from other instances in a different security group. As the owner of instances you can grant or revoke specific permissions using the [AuthorizeSecurityGroupIngress](#) and [RevokeSecurityGroupIngress](#) operations.

## Request Parameters

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

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

## Response Elements

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

Name	Description
<i>returnx</i>	true if call succeeded. Type: xsd:boolean

## Sample Request

```
<CreateSecurityGroup xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
  <groupName>WebServers</groupName>
  <groupDescription>Web</groupDescription>
</CreateSecurityGroup>
```

## Sample Response

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

## Related Operations

- [RunInstances](#)
- [DescribeSecurityGroups](#)
- [AuthorizeSecurityGroupIngress](#)



- [RevokeSecurityGroupIngress](#)
- [DeleteSecurityGroup](#)

## DeleteKeyPair

The `DeleteKeyPair` operation deletes a key pair.

### Request Parameters

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

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

### Response Elements

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

Name	Description
<code>return</code>	true if the key was successfully deleted. Type: xsd:boolean

### Sample Request

```
<DeleteKeyPair xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
  <keyName>example-key-name</keyName>
</DeleteKeyPair>
```

### Sample Response

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

### Related Operations

- [CreateKeyPair](#)
- [DescribeKeyPairs](#)

## DeleteSecurityGroup

The `DeleteSecurityGroup` operation deletes a security group.



#### Note

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

## Request Parameters

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

Name	Description	Required
<i>groupName</i>	Name of the security group to delete. Type: <code>xsd:string</code>	Yes

## Response Elements

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

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

## Sample Request

```
<DeleteSecurityGroup xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
  <groupName>RangedPortsBySource</groupName>
</DeleteSecurityGroup>
```

## Sample Response

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

## Related Operations

- [CreateSecurityGroup](#)
- [DescribeSecurityGroups](#)
- [AuthorizeSecurityGroupIngress](#)
- [RevokeSecurityGroupIngress](#)

## DeregisterImage

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

## Request Parameters

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

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

## Response Elements

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

Name	Description
<i>return</i>	true if deregistration succeeded; otherwise false. Type: xsd:boolean

## Sample Request

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

## Sample Response

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

## Related Operations

- [RegisterImage](#)
- [DescribeImages](#)

## DescribeImageAttribute

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

## Request Parameters

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

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

## Response Elements

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

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

## Sample Request - Launch Permission

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

## Sample Response - Launch Permission

```
<DescribeImageAttributeResponse xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
  <imageId>ami-61a54008</imageId>
  <launchPermission>
    <item>
      <group>all</group>
    </item>
    <item>
      <userId>495219933132</userId>
    </item>
  </launchPermission>
```

```
</DescribeImageAttributeResponse>
```

## Sample Request - Product Codes

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

## Sample Response - Product Codes

```
<DescribeImageAttributeResponse xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
  <imageId>ami-61a54008</imageId>
  <productCodes>
    <item>
      <productCode>774F4FF8</productCode>
    </item>
  </productCodes>
</DescribeImageAttributeResponse>
```

## Related Operations

- [DescribeImages](#)
- [ModifyImageAttribute](#)
- [ResetImageAttribute](#)

## DescribeImages

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

Launch permissions fall into three categories:

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

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

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

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

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



#### Note

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

## Request Parameters

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

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

## Response Elements

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

Name	Description
<i>imagesSet</i>	A list of image descriptions Type: <code>ec2:DescribeImagesResponseType[]</code>

## Sample Request

```
<DescribeImages xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
  <executableBySet>
    <item>
      <user>all</user>
    </item>
  </executableBySet>
  <ownersSet />
  <imagesSet>
    <item>
      <imageId>ami-61a54008</imageId>
      <imageId>ami-72f53012</imageId>
    </item>
  </imagesSet>
</DescribeImages>
```

```
</imagesSet>  
</DescribeImages>
```

## Sample Response

```
<DescribeImagesResponse xmlns="http://ec2.amazonaws.com/doc/2007-08-29">  
  <imagesSet>  
    <item>  
      <imageId>ami-61a54008</imageId>  
      <imageLocation>aes-ttylinux/image.manifest.xml</imageLocation>  
      <imageState>available</imageState>  
      <imageOwnerId>UYY3TLBUXIEON5NQVUUX6OMPWBZIQNFM</imageOwnerId>  
      <isPublic>true</isPublic>  
      <productCodes>  
        <item>  
          <productCode>774F4FF8</productCode>  
        </item>  
      </productCodes>  
    </item>  
  </imagesSet>  
</DescribeImagesResponse>
```

## Related Operations

- [DescribeInstances](#)
- [DescribeImageAttribute](#)

## DescribeInstances

The `DescribeInstances` operation returns information about instances that you own.

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

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

## Request Parameters

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

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

## Response Elements

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

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

## Sample Request

```
<DescribeInstances xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
  <instancesSet>
    <item>
      <instanceId>i-28a64341</instanceId>
    </item>
  </instancesSet>
</DescribeInstances>
```

## Sample Response

```
<DescribeInstancesResponse xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
  <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>
          <privateDnsName>domU-12-31-35-00-1E-01.compute-1.internal</privateDnsName>
          <dnsName>ec2-72-44-33-4.compute-1.amazonaws.com</dnsName>
          <keyName>example-key-name</keyName>
          <amiLaunchIndex>23</amiLaunchIndex>
          <productCodesSet>
            <item><productCode>774F4FF8</productCode></item>
          </productCodesSet>
          <instanceType>m1.large</instanceType>
          <launchTime>2007-08-07T11:54:42.000Z</launchTime>
        </item>
      </instancesSet>
    </item>
  </reservationSet>
</DescribeInstancesResponse>
```

## Related Operations

- [RunInstances](#)
- [TerminateInstances](#)

## DescribeKeyPairs



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

## Request Parameters

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

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

## Response Elements

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

Name	Description
<i>keySet</i>	A list of key pair descriptions Type: <a href="#">ec2:DescribeKeypairsResponseItemType[]</a>

## Sample Request

```
<DescribeKeyPairs xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
  <keySet>
    <item>
      <keyName>example-key-name</keyName>
    </item>
  </keySet>
</DescribeKeyPairs>
```

## Sample Response

```
<DescribeKeyPairsResponse xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
  <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](#)

## DescribeSecurityGroups

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

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

### Request Parameters

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

Name	Description	Required
<i>securityGroupSet</i>	List of security groups to describe. Type: <code>xsd:string[]</code>	Yes

### Response Elements

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

Name	Description
<i>securityGroupInfo</i>	Information about security groups. Type: <code>ec2:SecurityGroupItemType[]</code>

### Sample Request

```
<DescribeSecurityGroups xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
  <securityGroupSet>
    <item>
      <groupName>WebServers</groupName>
    </item>
    <item>
      <groupName>RangedPortsBySource</groupName>
    </item>
  </securityGroupSet>
</DescribeSecurityGroups>
```

### Sample Response

```
<DescribeSecurityGroupsResponse xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
  <securityGroupInfo>
    <item>
      <ownerId>UY3TLBUXIEON5NQVUUX6OMPWBZIQNFM</ownerId>
      <groupName>WebServers</groupName>
      <groupDescription>Web</groupDescription>
      <ipPermissions>
        <item>
          <ipProtocol>tcp</ipProtocol>
          <fromPort>80</fromPort>
          <toPort>80</toPort>
        </item>
      </ipPermissions>
    </item>
  </securityGroupInfo>
</DescribeSecurityGroupsResponse>
```

```
<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](#)

## GetConsoleOutput

The `GetConsoleOutput` operation retrieves console output for the specified instance.

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

## Request Parameters

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

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

## Response Elements

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

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

## Sample Request

```
<GetConsoleOutput xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
  <instanceId>i-28a64341</instanceId>
</GetConsoleOutput>
```

### Sample Response

```
<GetConsoleOutputResponse xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
  <instanceId>i-28a64341</instanceId>
  <timestamp>2007-01-03 15:00:00</timestamp>
  <output>TGludXggdmVyc2lubiAyLjYumTYteGVuVSAAoYnVpbGRlckBwYXRjaGJhdC5hbWFWf6b25zY
SkgKGdj
YyB2ZXJzaW9uIDQuMC4xIDIwMDUwNzI3IChSZSWQgSGF0IDQuMC4xLTUpKSAjMSBTTVAgVGhlIE9j
dCAyNiAwODo0MToyNiBTRGVNUIDiWMdYkQklPUy1wcm92aWR1ZCBwaHlzawNhCBSQU0gbWFwOgpY
ZWA6IDRUMDADMdAWMDADMdAGLSAwmADwMDAwMDZhNDAwMDAwIChlc2FibGUpCjk4MElCEhY
R0hnRU0gYXZhZWxhYm91Lgo3MjdndQk1BMTl0gYXZhZWxhYm91LgpoWC0ARXhlY3V0ZSBGAeXNh
Ymx1KSBCwcm90ZWNoaW9uOiBhY3RpdmlUKSVJRIGxvY2t1cCBkZXRLY3Rpb24gZGlzYWJsZWQKQnVp
bHQgMSB6b25lbG1zdHMKS2VybmVsIGNvbWlhbm9qbGluZTogcm9vdD0vZGVZ2L3NkYTEgcmlgNApF
bmFibGluZyBmYXN0IEZQVSBzYXZlIGFuZCBYXXN0b3JlLi4uIGRvbmUuCg==</output>
</GetConsoleOutputResponse>
```

## ModifyImageAttribute

The `ModifyImageAttribute` operation modifies an attribute of an AMI.

## Request Parameters

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

Name	Description	Required
<i>imageId</i>	AMI ID to modify. Type: xsd:string	Yes
<i>launchPermission</i>	Adds or removes launch permissions for the AMI. Type: <a href="#">ec2:LaunchPermissionOperationType</a>	Choice
<i>productCodes</i>	Attaches a product code to the AMI, allowing developers to charge for the use of their AMIs. Currently only one product code can be associated with an AMI. Once set,	Choice

Name	Description	Required
	the product code can not be changed or reset. Type: <a href="#">ec2:ProductCodeItemType[]</a>	

## Response Elements

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

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

## Sample Request - Launch Permission

```
<ModifyImageAttribute xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
  <imageId>ami-61a54008</imageId>
  <launchPermission>
    <add>
      <item>
        <group>all</group>
      </item>
      <item>
        <userId>495219933132</userId>
      </item>
    </add>
  </launchPermission>
</ModifyImageAttribute>
```

## Sample Request - Product Codes

```
<ModifyImageAttribute xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
  <imageId>ami-61a54008</imageId>
  <productCodes>
    <item>
      <productCode>774F4FF8</productCode>
    </item>
  </productCodes>
</ModifyImageAttribute>
```

## Sample Response

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

## Related Operations

- [ResetImageAttribute](#)

- [DescribeImageAttribute](#)

## RebootInstances

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

### Request Parameters

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

Name	Description	Required
<i>instancesSet</i>	One or more instance IDs. Type: <code>xsd:string[]</code>	Yes

### Response Elements

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

Name	Description
<i>result</i>	true if the operation succeeded. Type: <code>xsd:boolean</code>

### Sample Request

```
<RebootInstances xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
  <instancesSet>
    <item>
      <instanceId>i-28a64341</instanceId>
    </item>
  </instancesSet>
</RebootInstances>
```

### Sample Response

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

## RegisterImage

The `RegisterImage` operation registers an AMI with Amazon EC2. Images must be registered before they can be [launched](#).

Each AMI is associated with a unique ID which is provided by the Amazon EC2 service through the `RegisterImage` operation. During registration, Amazon EC2 retrieves the specified image manifest

from Amazon S3 and verifies that the image is owned by the user registering the image.

The image manifest is retrieved once and stored within the Amazon EC2. Any modifications to an image in Amazon S3 invalidates this registration. If you make changes to an image, [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.

Name	Description	Required
<i>imageLocation</i>	Full path to your AMI manifest in Amazon S3 storage. Type: xsd:string	Yes

## Response Elements

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

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

## Sample Request

```
<RegisterImage xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
  <imageLocation>/mybucket/myimage.manifest.xml</imageLocation>
</RegisterImage>
```

## Sample Response

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

## Related Operations

- [DescribeImages](#)
- [DeregisterImage](#)

## ResetImageAttribute

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



### Note

The productCodes attribute cannot be reset.

## Request Parameters

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

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

## Response Elements

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

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

## Sample Request

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

## Sample Response

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

## Related Operations

- [ModifyImageAttribute](#)
- [DescribeImageAttribute](#)

## RevokeSecurityGroupIngress

The `RevokeSecurityGroupIngress` operation revokes permissions from a security group. The



permissions used to revoke must be specified using the same values used to grant the permissions.

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

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

## Request Parameters

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

Name	Description	Required
<i>userId</i>	AWS Access Key ID. Type: xsd:string	Yes
<i>groupName</i>	Name of the group to modify. Type: xsd:string	Yes
<i>ipPermissions</i>	Set of permissions to remove from the group. Type: <a href="#">ec2:IpPermissionType</a> []	Yes

## Response Elements

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

Name	Description
<code>return</code>	true if permissions successfully revoked. Type: xsd:boolean

## Sample Request

```
<RevokeSecurityGroupIngress xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
  <userId/>
  <groupName>RangedPortsBySource</groupName>
  <ipPermissions>
    <item>
      <ipProtocol>tcp</ipProtocol>
      <fromPort>6000</fromPort>
      <toPort>7000</toPort>
      <groups/>
      <ipRanges/>
    </item>
  </ipPermissions>
</RevokeSecurityGroupIngress>
```

## Sample Response

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

## Related Operations

- [CreateSecurityGroup](#)
- [DescribeSecurityGroups](#)
- [AuthorizeSecurityGroupIngress](#)
- [DeleteSecurityGroup](#)

## RunInstances

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

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

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

1. Libby runs the `RunInstances` operation to launch database instances (min. 10, max. 15) and web\_server instances (min. 30, max. 40).

Because there are currently 30 instances available and Libby needs a minimum of 40, no instances are launched.

2. Libby adjusts the number of instances she needs and runs the `RunInstances` operation to launch database instances (min. 5, max. 10) and web\_server instances (min. 20, max. 40).

Amazon EC2 launches the minimum number of instances for each AMI (5 database, 20 web\_server).

The remaining 5 instances are allocated using round robin.

3. Libby adjusts the number of instances she needs and runs the `RunInstances` operation again to launch database instances (min. 5, max. 10) and web\_server instances (min. 20, max. 40).



### Note

Every instance is launched in a [security group](#). If you do not specify a security group at launch, the instances start in your default security group.

You can provide an optional [key pair ID](#) for each image in the launch request. All instances that are created from images that use this key pair will have access to the associated public key at boot. You can use this key to provide secure access to an instance of an image on a per-instance basis. Amazon EC2 public images use this feature to provide secure access without passwords.



### Important

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

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

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




**Note**

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

## Request Parameters

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

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

## Response Elements

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

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

## Sample Request

```
<RunInstances xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
  <instancesSet>
    <item>
      <imageId>ami-60a54009</imageId>
      <minCount>1</minCount>
      <maxCount>3</maxCount>
      <keyName>example-key-name</keyName>
    </item>
  </instancesSet>
  <groupSet/>
  <userData version="1.0" encoding="base64"><data>VGhpcyBpcyBiYXNlIDY0IQ==</data></userData>
  <addressingType>public</addressingType>
</RunInstances>
```

## Sample Response

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

```
<amiLaunchIndex>1</amiLaunchIndex>
<instanceType>m1.small</instanceType>
<launchTime>2007-08-07T11:51:50.000Z</launchTime>
</item>
<item>
  <instanceId>i-2be64332</instanceId>
  <imageId>ami-60a54009</imageId>
  <instanceState>
    <code>0</code>
    <name>pending</name>
  </instanceState>
  <privateDnsName></privateDnsName>
  <dnsName></dnsName>
  <keyName>example-key-name</keyName>
  <amiLaunchIndex>2</amiLaunchIndex>
  <instanceType>m1.small</instanceType>
  <launchTime>2007-08-07T11:51:50.000Z</launchTime>
</item>
</instancesSet>
</RunInstancesResponse>
```

## Related Operations

- [DescribeInstances](#)
- [TerminateInstances](#)
- [AuthorizeSecurityGroupIngress](#)
- [RevokeSecurityGroupIngress](#)
- [DescribeSecurityGroups](#)

## TerminateInstances

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

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

## Request Parameters

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

Name	Description	Required
<i>instancesSet</i>	One or more instance IDs. Type: <code>xsd:string[]</code>	Yes

## Response Elements

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

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

Name	Description

## Sample Request

```
<TerminateInstances xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
  <instancesSet>
    <item>
      <instanceId>i-28a64341</instanceId>
    </item>
  </instancesSet>
</TerminateInstances>
```

## Sample Response

```
<TerminateInstancesResponse xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
  <instancesSet>
    <item>
      <instanceId>i-28a64341</instanceId>
      <shutdownState>
        <code>32</code>
        <name>shutting-down</name>
      </shutdownState>
      <previousState>
        <code>16</code>
        <name>running</name>
      </previousState>
    </item>
  </instancesSet>
</TerminateInstancesResponse>
```

## Related Operations

- [DescribeInstances](#)

# Amazon EC2 Query API

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

## Common Query Parameters

### Request Parameters

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

Name	Description	Required
<i>Action</i>	Indicates the action to perform. Example: RunInstances	Yes
<i>Version</i>	The API version to use, as specified in the WSDL. Example: 2007-08-29	Yes

API Version 2007-08-29

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



#### Note

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

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

## List of Operations by Function

### Images

- [RegisterImage](#)
- [DescribeImages](#)
- [DeregisterImage](#)

### Instances

- [RunInstances](#)
- [DescribeInstances](#)
- [TerminateInstances](#)
- [ConfirmProductInstance](#)

#### Key Pairs

- [CreateKeyPair](#)
- [DescribeKeyPairs](#)
- [DeleteKeyPair](#)

#### Image Attributes

- [ModifyImageAttribute](#)
- [DescribeImageAttribute](#)
- [ResetImageAttribute](#)

#### Security Groups

- [CreateSecurityGroup](#)
- [DescribeSecurityGroups](#)
- [DeleteSecurityGroup](#)
- [AuthorizeSecurityGroupIngress](#)
- [RevokeSecurityGroupIngress](#)

## AuthorizeSecurityGroupIngress

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

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

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

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

## Request Parameters

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



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

## Response Elements

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

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

## Sample Request

```
https://ec2.amazonaws.com/
?Action=AuthorizeSecurityGroupIngress
&IpProtocol=tcp
```

```
&FromPort=80
&ToPort=80
&CidrIp=0.0.0.0/0
&...auth parameters...
```

## Sample Response

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

## Related Operations

- [CreateSecurityGroup](#)
- [DescribeSecurityGroups](#)
- [RevokeSecurityGroupIngress](#)
- [DeleteSecurityGroup](#)

## ConfirmProductInstance

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

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

## Request Parameters

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

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

## Response Elements

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

Name	Description
result	true if the product code is attached to the instance, false if it is not. Type: xsd:boolean

Name	Description
ownerId	The instance owner's account ID. Only present if the product code is attached to the instance. Type: xsd:string

## Sample Request

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

## Sample Response

```
<ConfirmProductInstanceResponse xm  
lns="http://ec2.amazonaws.com/doc/2007-08-29">  
  <result>true</result>  
  <ownerId>254933287430</ownerId>  
</ConfirmProductInstanceResponse>
```

## Related Operations

- [DescribeInstances](#)
- [RunInstances](#)

## CreateKeyPair

The `CreateKeyPair` operation creates a new 2048 bit RSA key pair and returns a unique ID that can be used to reference this key pair when [launching](#) new instances.

## Request Parameters

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

Name	Description	Required
<i>KeyName</i>	A unique name for the key pair. Type: String	Yes

## Response Elements

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

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

## Sample Request

```
https://ec2.amazonaws.com/
?Action=CreateKeyPair
&KeyName=example-key-name
&...auth parameters...
```

## Sample Response

```
<CreateKeyPairResponse xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
  <keyName>example-key-name</keyName>
  <keyFingerprint>1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:f1:6f</
  keyFingerprint>
  <keyMaterial>-----BEGIN RSA PRIVATE KEY-----
MIIIEoQIBAAKCAQBULFg5ujHrtmljnutSuoO8Xe56LlT+HM8v/xkaa39EstM3/aFXTgElQiJLChp
HungXQ29VTc8rc1bw0lkdi23OH5eqkMHGhvEwqa0HWASUM114o3o/IX+0f2UcPoKCOVUR+jx7lSg
5AU52EQfanIn3ZQ8lFW7Edp5a3q4DhJG1UKToHVbicL5E+g45zfB95wIyywWZfEW/UUF3LpGZyq/
ebiU1qlqTbHkLbCC2r7RTn8vpQWp47BGVYgtGSBmPTRP5hnbzzuqj3itkiLHjU39S2sJCJ0TrJx5
i8BygR4s3mHKbj8l+ePQxG1kGbF6R4yg6sECmXn17MRQVXODNHZbAgMBAAECggEAY1tsiUsIwD15
91CXirkYGuVfLyLflXenxfI50mDFms/mumTqloHO7tr0oriHDR5K7wMcY/YY5YkcXNo7mvUVD1pM
ZNUJs7rw9gZRTTrf7LylaJ58kOcyajw8TsC4e4LPbFaHwSld6K8rXh64o6WgW4SrsB6ICmr1kGQI7
3wcfgt5ecIu4TZf00E9IHjn+2eRlsrjBdeORi7KiUNC/pAG23I6MdDOFEQRcCSigCj+4/mciFUSA
SWS4dMbrpb9FNSIcf9dcLxVM7/6KxgJNfZc9XWzUw77Jg8x92Zd0fVhH0ux5IZC+UvSKWB4dyfcI
tE8C3p9bbU9VGyY5vLCAiIb4qQKBgQDLiO24GXrIkswF32YtBBMuVgLGcWU9h9H1O9mKAc2m8Cm1
jUE5IpzRjTedc9I2qiIMUTwtgnw42auSCzbUeYMURPtDqyQ7p6A jMu jp9EPemcSVOK9vXYL0Ptco
xW9MC0dtV6iPkCN7gOqiZXPkKaFbWADp16p8UAIVS/a5XXk5jwKBgQCKkpHi2EIShlurKxhljyWC
iDCiK6JBRsMvpLbc0v5dKwP5alo1fmdR5PJaV2qvZSj5CYNpMay1/EDNTY5OSIJU+OKFmQbyhsbm
rdLNLDL4+TcnT7c62/aH0lohYaf/VCbRhtLlBfGGoQc7+sAc8vmKkesnF7CqCEKDyF/dhrxYdQKB
gC0iZzzNAapayz1+JcVTWwEid6j9JqNXbBc+Z2YwMi+T0Fv/P/hwkX/ypeOXnIUcw0Ih/YtGBVAC
DQbsz7LcYlHqXiHKYNWNvXgwwO+oiChjxvEkSdsTTIfnK4VSCvU9BxDbQHjdiNDJbL6oar92UN7V
rBYvChJZF7LvUH4YmVpHAoGAbZ2X7XvooEO+uZ58/BGKOIGHByHBDiXtzmhdJr15HTYjxK7OgTZm
gK+8zp4L9IbVLGDMJO8vft32XPEWuvI8twCzFH+CsWLQADZMZKSSBasOZ/h1FwhdMgCMcY+Qlzd4
JZKjTSu3i7vhvx6RzdSedXEMNTZWN4qlIx3kR5aHcukCgYA9T+ZrvmlF0seQPbLknn7EqhXIjBaT
P8TTvW/6bdPi23ExzxZn7K0drfclyRphlLHMPaONv/x2xALIf91UB+v5ohyloDoasL0gi jlhore
2ERKKdwz0ZL9SWq6VTdhr/5G994CK72fy5WhyERBDjUIDHaK3M849JJuf8cSrvSb4g==
-----END RSA PRIVATE KEY-----</keyMaterial>
</CreateKeyPairResponse>
```

## Related Operations

- [DescribeKeyPairs](#)
- [DeleteKeyPair](#)
- [RunInstances](#)

## CreateSecurityGroup

The CreateSecurityGroup operation creates a new security group.

Every instance is launched in a security group. If no security group is specified during launch, the instances are launched in the default security group. Instances within the same security group have unrestricted network access to each other. Instances will reject network access attempts from other instances in a different security group. As the owner of instances you can grant or revoke specific permissions using the [AuthorizeSecurityGroupIngress](#) and [RevokeSecurityGroupIngress](#) operations.

### Request Parameters

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

Name	Description	Required
<i>GroupName</i>	Name of the new security group. Type: String	Yes
<i>GroupDescription</i>	Description of the new security group. Type: String	Yes

### Response Elements

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

Name	Description
<code>return</code>	true if call succeeded. Type: xsd:boolean

### Sample Request

```
https://ec2.amazonaws.com/  
?Action==CreateSecurityGroup  
&GroupName=WebServers  
&GroupDescription=Web  
&...auth parameters...
```

### Sample Response

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

### Related Operations

- [RunInstances](#)
- [DescribeSecurityGroups](#)
- [AuthorizeSecurityGroupIngress](#)

- [RevokeSecurityGroupIngress](#)
- [DeleteSecurityGroup](#)

## DeleteKeyPair

The DeleteKeyPair operation deletes a key pair.

### Request Parameters

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

Name	Description	Required
<i>KeyName</i>	Name of the key pair to delete. Type: String	Yes

### Response Elements

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

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

### Sample Request

```
https://ec2.amazonaws.com/  
?Action=DeleteKeyPair  
&KeyName=example-key-name  
&...auth parameters...
```

### Sample Response

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

### Related Operations

- [CreateKeyPair](#)
- [DescribeKeyPairs](#)

## DeleteSecurityGroup

The DeleteSecurityGroup operation deletes a security group.



#### Note

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

## Request Parameters

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



#### Note

A security group cannot be deleted if it is referenced by another security group. For example, if security group B has a rule that allows access from security group A, security group A cannot be deleted until the allow rule is removed.

Name	Description	Required
<i>GroupName</i>	Name of the security group to delete. Type: String	Yes

## Response Elements

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

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

## Sample Request

```
https://ec2.amazonaws.com/  
?Action=DeleteSecurityGroup  
&GroupName=RangedPortsBySource  
&...auth parameters...
```

## Sample Response

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

## Related Operations

- [CreateSecurityGroup](#)

- [DescribeSecurityGroups](#)
- [AuthorizeSecurityGroupIngress](#)
- [RevokeSecurityGroupIngress](#)

## DeregisterImage

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

### Request Parameters

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

Name	Description	Required
<i>ImageId</i>	Unique ID of a machine image, returned by a call to <a href="#">RegisterImage</a> or <a href="#">DescribeImages</a> . Type: String	Yes

### Response Tags

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

Name	Description
<code>return</code>	true if deregistration succeeded; otherwise false. Type: xsd:boolean

### Sample Request

```
https://ec2.amazonaws.com/  
?Action=DeregisterImage  
&ImageId=ami-61a54008  
&...auth parameters...
```

### Sample Response

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

### Related Operations

- [RegisterImage](#)
- [DescribeImages](#)

## DescribeImageAttribute



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

## Request Parameters

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

Name	Description	Required
<code>ImageIdParam;</code>	ID of the AMI for which an attribute will be described. Type: String	Yes
<i>Attribute</i>	Specifies the attribute to describe. Type: String Valid Value: <code>launchPermission</code>	Yes

## Attributes

Attribute Name	Description
<code>launchPermission</code>	The AMIs launch permissions.
<code>productCodes</code>	The product code attached to the AMI.

## Response Elements

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

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

## Sample Request - Launch Permission

```
https://ec2.amazonaws.com/  
?Action=DescribeImageAttribute  
&ImageId=ami-61a54008  
&Attribute=launchPermission  
&...auth parameters...
```

## Sample Response - Launch Permission

```
<DescribeImageAttributeResponse xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
  <imageId>ami-61a54008</imageId>
  <launchPermission>
    <item>
      <group>all</group>
    </item>
    <item>
      <userId>495219933132</userId>
    </item>
  </launchPermission>
</DescribeImageAttributeResponse>
```

## Sample Request - Product Codes

```
https://ec2.amazonaws.com/
?Action=DescribeImageAttribute
&ImageId=ami-61a54008
&Attribute=productCodes
&...auth parameters...
```

## Sample Response - Product Codes

```
<DescribeImageAttributeResponse xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
  <imageId>ami-61a54008</imageId>
  <productCodes>
    <item>
      <productCode>774F4FF8</productCode>
    </item>
  </productCodes>
</DescribeImageAttributeResponse>
```

## Related Operations

- [DescribeImages](#)
- [ModifyImageAttribute](#)
- [ResetImageAttribute](#)

## DescribeImages

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

Launch permissions fall into three categories:

Launch Permission	Description
public	The owner of the AMI granted launch permissions for the AMI to the <code>all</code> group. All users have launch permissions for these AMIs.

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

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

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

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

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



#### Note

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

## Request Parameters

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

Name	Description	Required
<i>ImageId.n</i>	A list of image descriptions Type: String	No
<i>Owner.n</i>	Owners of AMIs to describe. Type: String	No
<i>ExecutableBy.n</i>	AMIs for which specified users have access. Type: String	No

## Response Elements

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

Name	Description
<i>imagesSet</i>	A list of image descriptions. Type: <a href="#">ec2:DescribeImagesResponseType</a> []

Name	Description

## Sample Request

```
https://ec2.amazonaws.com/  
?Action=DescribeImages  
&ImageId.1=ami-61a54008  
&...auth parameters...
```

## Sample Response

```
<DescribeImagesResponse xmlns="http://ec2.amazonaws.com/doc/2007-08-29">  
  <imagesSet>  
    <item>  
      <imageId>ami-61a54008</imageId>  
      <imageLocation>aes-ttylinux/image.manifest.xml</imageLocation>  
      <imageState>available</imageState>  
      <imageOwnerId>UYY3TLBUXIEON5NQVUUX6OMPWBZIQNFM</imageOwnerId>  
      <isPublic>>false</isPublic>  
      <productCodes>  
        <item>  
          <productCode>774F4FF8</productCode>  
        </item>  
      </productCodes>  
    </item>  
  </imagesSet>  
</DescribeImagesResponse>
```

## Related Operations

- [DescribeInstances](#)
- [DescribeImageAttribute](#)

## DescribeInstances

The `DescribeInstances` operation returns information about instances that you own.

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

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

## Request Parameters

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

Name	Description	Required
<i>InstanceId.n</i>	Set of instances IDs to get the status of. Type: String	No

API Version 2007-08-29

Name	Description	Required

## Response Elements

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

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

## Sample Request

```
https://ec2.amazonaws.com/  
?Action=DescribeInstances  
&InstanceId.1=i-28a64341  
&...auth parameters...
```

## Sample Response

```
<DescribeInstancesResponse xmlns="http://ec2.amazonaws.com/doc/2007-08-29">  
  <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>  
          <privateDnsName>domU-12-31-35-00-1E-01.compute-1.internal</privateDnsName>  
          <dnsName>ec2-72-44-33-4.compute-1.amazonaws.com</dnsName>  
          <keyName>example-key-name</keyName>  
          <productCodesSet>  
            <item><productCode>774F4FF8</productCode></item>  
          </productCodesSet>  
          <InstanceType>m1.small</InstanceType>  
          <launchTime>2007-08-07T11:54:42.000Z</launchTime>  
        </item>  
      </instancesSet>  
    </item>  
  </reservationSet>  
</DescribeInstancesResponse>
```

## Related Operations

- [RunInstances](#)
- [TerminateInstances](#)

## DescribeKeyPairs

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

### Request Parameters

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

Name	Description	Required
<i>KeyName</i> . <i>n</i>	Key pair IDs to describe. Type: String	No

### Response Elements

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

Name	Description
keySet	A list of key pair descriptions. Type: <a href="#">ec2:DescribeKeypairsResponseType</a> []

### Sample Request

```
https://ec2.amazonaws.com/  
?Action=DescribeKeyPairs  
&KeyName.1=example-key-name  
&...auth parameters...
```

### Sample Response

```
<DescribeKeyPairsResponse xmlns="http://ec2.amazonaws.com/doc/2007-08-29">  
  <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](#)

## DescribeSecurityGroups

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

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

### Request Parameters

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

Name	Description	Required
<i>GroupName.n</i>	List of security groups to describe. Type: String	No

### Response Elements

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

Name	Description
securityGroupInfo	Information about security groups. Type: <a href="#">ec2:SecurityGroupItemType</a> []

### Sample Request

```
https://ec2.amazonaws.com/  
?Action=DescribeSecurityGroups  
&GroupName.1=WebServers  
&GroupName.2=RangedPortsBySource  
&...auth parameters...
```

### Sample Response

```
<DescribeSecurityGroupsResponse xmlns="http://ec2.amazonaws.com/doc/2007-08-29">  
  <securityGroupInfo>  
    <item>  
      <ownerId>UYY3TLBUXIEON5NQVUUX6OMPWBZIQNFM</ownerId>  
      <groupName>WebServers</groupName>  
      <groupDescription>Web</groupDescription>  
      <ipPermissions>  
        <item>  
          <ipProtocol>tcp</ipProtocol>  
          <fromPort>80</fromPort>  
          <toPort>80</toPort>  
        </item>  
      </ipPermissions>  
      <ipRanges>  
        <item>  
          <cidrIp>0.0.0.0/0</cidrIp>  
        </item>  
      </ipRanges>  
    </item>  
  </securityGroupInfo>  
</DescribeSecurityGroupsResponse>
```

```
</item>
</ipPermissions>
</item>
<item>
  <ownerId>UY3TLBUXIEON5NQVUUX6OMPWBZIQNFM</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](#)

## GetConsoleOutput

The `GetConsoleOutput` operation retrieves console output for the specified instance.

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

## Request Parameters

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

Name	Description	Required
<i>InstanceId</i>	An instance ID returned from a previous call to <code>RunInstances</code> . Type: String	Yes

## Response Elements

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

Name	Description
<code>instanceId</code>	The instance ID. Type: <code>xsd:string</code>



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

## Sample Request

```
https://ec2.amazonaws.com/  
?Action=GetConsoleOutput  
&InstanceId.1=i-2ea64347  
&...auth parameters...
```

## Sample Response

```
<GetConsoleOutputResponse xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
  <instanceId>i-28a64341</instanceId>
  <timestamp>2007-01-03 15:00:00</timestamp>
  <output>TGludXggdmVyc2lubiAyLjYumTYteGVuVS AoYnVpbGRlc kBWYXRjaGJhdC5hbW F6b25zY
SkgKGdj
YyB2ZXJzaW9uIDQuMC4xIDIwMDUwNzI3IChSZWQgSGF0IDQuMC4xLTUpKSAjMSBTTVAgVGhlIE9j
dCAyNiAwODo0MT0yNiBTQUVUIDIwMDYkYkdlPUYlwcm92aWR1ZCBwaHlzcWNhbCBSQU0gbWFwOgpY
ZWA6IDAwdAwdMDAwMDAwMDAwMDAGLSAwMDAwMDAwMDZhndAwdMAwIChlc2FibG9uPjk4MElCEhJ
R0hNRU0gYXZhaWxhYmx1LGo3MjdjNDQ1BMTldnRU0gyXZhaWxhYmx1L GpOWCAoRXhlyY3V0ZSB EAeXNh
Ymx1KS Bwc m90ZWNoaW9uOi BhY3RpdmUKSVJRIGxvY2t1cCBkZXRLY3Rpb24gZGlzYWJsZW QKQ nVp
bHQgMSB6b25lbGlzdHMKS2VybmVsIGNvbWl hbmQybGl uZTogcm9vdD0vZGVZ2L3NkYTEgc m8gNApF
bmFibGluZyBmYXN0IEZQVSBZXXZlIGFuZCBY ZXN0b3JlLi4uIGRvbmUuCg==</output>
</GetConsoleOutputResponse>
```

## ModifyImageAttribute

The `ModifyImageAttribute` operation modifies an attribute of an AMI.

## Attributes

Attribute Name	Type	Description
launchPermission	List	Controls who has permission to launch the AMI. Launch permissions can be granted to specific users by adding userIds. To make the AMI public, add the all group.
productCodes	List	Associates a product code with AMIs. This allows developers to charge users for using AMIs. The user must be signed up for the product before they can launch the AMI. This is a write once attribute; after it is set, it cannot be changed or removed.

## Request Parameters

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

Name	Description	Required
<i>ImageId</i>	AMI ID to modify. Type: String	Yes
<i>Attribute</i>	Specifies the attribute to modify. See the preceding attributes table for supported attributes. Type: String	Yes
<i>OperationType</i>	Specifies the operation to perform on the attribute. See the preceding attributes table for supported operations for attributes. Type: String Valid Values: add   remove Condition: Required for launchPermission	Conditional
<i>UserId.n</i>	User IDs to add to or remove from the launchPermission attribute. Type: String Condition: Required for launchPermission	Conditional
<i>UserGroup.n</i>	User groups to add to or remove from the launchPermission attribute. Currently, the all group is available, which will make it a public AMI. Type: String Condition: Required for launchPermission	Conditional
<i>ProductCode.n</i>	Attaches a product code to the AMI. Currently only one product code can be associated with an AMI. Once set, the product code cannot be changed or reset. Type: String Condition: Required for productCodes	Conditional

## Response Elements

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

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

## Sample Request - Launch Permission

```
https://ec2.amazonaws.com/  
?Action=ModifyImageAttribute  
&ImageId=ami-61a54008  
&Attribute=launchPermission  
&OperationType=add  
&Group.1=all  
&UserId.1=495219933132  
&...auth parameters...
```

## Sample Request - Product Codes

```
https://ec2.amazonaws.com/  
?Action=ModifyImageAttribute  
&ImageId=ami-61a54008  
&Attribute=productCodes  
&ProductCode.1=774F4FF8  
&...auth parameters...
```

## Sample Response

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

## Related Operations

- [ResetImageAttribute](#)
- [DescribeImageAttribute](#)

## RebootInstances

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

## Request Parameters

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

Name	Description	Required
<i>InstanceId.n</i>	One or more instance IDs. Type: String	Yes

## Response Elements

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

Name	Description
result	true if the operation succeeded.

Name	Description
	Type: xsd:boolean

## Sample Request

```
https://ec2.amazonaws.com/  
?Action=RebootInstances  
&InstanceId.1=i-2ea64347  
&InstanceId.2=i-21a64348  
&...auth parameters...
```

## Sample Response

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

## RegisterImage

The `RegisterImage` operation registers an AMI with Amazon EC2. Images must be registered before they can be [launched](#).

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

The image manifest is retrieved once and stored within the Amazon EC2. Any modifications to an image in Amazon S3 invalidates this registration. If you make changes to an image, [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.

Name	Description	Required
<i>ImageLocation</i>	Full path to your AMI manifest in Amazon S3 storage. Type: String	Yes

## Response Elements

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

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

## Sample Request

```
https://ec2.amazonaws.com/  
?Action=RegisterImage  
&ImageLocation=mybucket-myimage.manifest.xml  
&...auth parameters...
```

## Sample Response

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

## Related Operations

- [DescribeImages](#)
- [DeregisterImage](#)

## ResetImageAttribute

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



### Note

The `productCodes` attribute cannot be reset.

## Request Parameters

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

Name	Description	Required
<i>ImageId</i>	ID of the AMI for which an attribute will be described. Type: String	Yes
<i>Attribute</i>	Specifies the attribute to reset. Currently, only <code>launchPermission</code> is supported. In the case of <code>launchPermission</code> , all public and explicit launch permissions for the AMI are revoked. Type: String	Yes

## Response Elements

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

Name	Description
<code>return</code>	true if the operation succeeded, otherwise false.

Name	Description
	Type: xsd:boolean

## Sample Request

```
https://ec2.amazonaws.com/  
?Action=ResetImageAttribute  
&ImageId=ami-61a54008  
&Attribute=launchPermission  
&...auth parameters...
```

## Sample Response

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

## Related Operations

- [ModifyImageAttribute](#)
- [DescribeImageAttribute](#)

## RevokeSecurityGroupIngress

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

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

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

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

## Request Parameters

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

Name	Description	Required
<i>GroupName</i>	Name of the group to modify. Type: String	Yes
<i>SourceSecurityGroupName</i>	Name of security group to revoke access to when operating on a user/group pair.	Conditional

Name	Description	Required
	Type: String Condition: Required when revoking user/group pair permission.	
<i>SourceSecurityGroupOwnerId</i>	Owner of security group to revoke access to when operating on a user/group pair. Type: String Condition: Required when revoking user/group pair permission.	Conditional
<i>IpProtocol</i>	IP protocol to revoke access to when operating on a CIDR IP. Type: String Valid Values: tcp   udp   icmp Condition: Required when revoking CIDR IP permission.	Conditional
<i>FromPort</i>	Bottom of port range to revoke access to when operating on a CIDR IP. This contains the ICMP type if ICMP is being authorized. Type: Int Condition: Required when revoking CIDR IP permission.	Conditional
<i>ToPort</i>	Top of port range to revoke access to when operating on a CIDR IP. This contains the ICMP code if ICMP is being authorized. Type: Int Condition: Required when revoking CIDR IP permission.	Conditional
<i>CidrIp</i>	CIDR IP range to revoke access to when operating on a CIDR IP. Type: String Condition: Required when revoking CIDR IP permission.	Conditional

## Response Elements

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

Name	Description
<code>return</code>	true if permissions successfully revoked. Type: xsd:boolean

## Sample Request

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

## Sample Response

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

## Related Operations

- [CreateSecurityGroup](#)
- [DescribeSecurityGroups](#)
- [AuthorizeSecurityGroupIngress](#)
- [DeleteSecurityGroup](#)

## RunInstances

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



### Note

The Query version of `RunInstances` only allows instances of a single AMI to be launched in one call. This is different from the SOAP API version of the call, but similar to the **ec2-run-instances** command line tool.

If Amazon EC2 cannot launch the minimum number AMIs you request, no instances launch. If there is insufficient capacity to launch the maximum number of AMIs you request, Amazon EC2 launches as many as possible to satisfy the requested maximum values.

Every instance is launched in a [security group](#). If you do not specify a security group at launch, the instances start in your default security group.

An optional instance type can be specified. For information about instance types, see [Selecting Instance Types](#).

You can provide an optional [key pair ID](#) for each image in the launch request. All instances that are created from images that use this key pair will have access to the associated public key at boot. You can use this key to provide secure access to an instance of an image on a per-instance basis. Amazon EC2 public images use this feature to provide secure access without passwords.



### Important

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



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

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




**Note**

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

## Request Parameters

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

Name	Description	Required
<i>ImageId</i>	ID of the AMI with which to launch instances. Type: String	Yes
<i>MinCount</i>	Minimum number of instances to launch. Type: Int	Yes
<i>MaxCount</i>	Maximum number of instances to launch. Type: Int	Yes
<i>KeyName</i>	Name of the key pair with which to launch instances. Type: String	No
<i>SecurityGroup.n</i>	Names of the security groups with which to associate the instances. Type: String	No
<i>UserData</i>	The user data available to the launched instances. This should be Base64 encoded. For more information on encoding details, see <a href="#">UserDataType</a> . Type: String	No
<i>AddressingType</i>	The addressing scheme with which to launch the instance. The supported addressing type is <code>public</code> . For the public scheme, the instance has a private and public IP address that are mapped through NAT. For more information, see <a href="#">Instance Addressing</a> .  <div>  <p><b>Note</b></p> <p>To support our legacy cluster, early beta customers can also specify <code>direct</code>. In the direct scheme, the instance has a single public IP address.</p> </div>	No
<b>API Version 2007-08-29</b>		

Name	Description	Required
	Type: String	
<i>InstanceType</i>	<p>This specifies the instance type.</p> <p>Options include <code>m1.small</code>, <code>m1.large</code>, and <code>m1.xlarge</code>.</p> <p>The default value is <code>m1.small</code>.</p> <p>For more information on instance types, see <a href="#">Selecting Instance Types</a></p> <p>Type: <code>xsd:string</code></p>	No

## Response Elements

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

Name	Description
RunInstancesResponse	<p>Status information about the instances launched.</p> <p>Type: <a href="#">ec2ReservationInfoType</a></p>

## Sample Request

```
https://ec2.amazonaws.com/
?Action=RunInstances
&ImageId=ami-60a54009
&MaxCount=3
&MinCount=1
&AddressingType=public
&...auth parameters...
```

## Sample Response

```
<RunInstancesResponse xmlns="http://ec2.amazonaws.com/doc/2007-08-29">
  <reservationId>r-47a5402e</reservationId>
  <ownerId>495219933132</ownerId>
  <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>
      <privateDnsName></privateDnsName>
      <dnsName></dnsName>
      <keyName>example-key-name</keyName>
      <amiLaunchIndex>0</amiLaunchIndex>
      <InstanceType>m1.small</InstanceType>
      <launchTime>2007-08-07T11:51:50.000Z</launchTime>
    </item>
    <item>
```

```
<instanceId>i-2bc64242</instanceId>
<imageId>ami-60a54009</imageId>
<instanceState>
  <code>0</code>
  <name>pending</name>
</instanceState>
<privateDnsName></privateDnsName>
<dnsName></dnsName>
<keyName>example-key-name</keyName>
<amiLaunchIndex>1</amiLaunchIndex>
<InstanceType>m1.small</InstanceType>
<launchTime>2007-08-07T11:51:50.000Z</launchTime>
</item>
<item>
  <instanceId>i-2be64332</instanceId>
  <imageId>ami-60a54009</imageId>
  <instanceState>
    <code>0</code>
    <name>pending</name>
  </instanceState>
  <privateDnsName></privateDnsName>
  <dnsName></dnsName>
  <keyName>example-key-name</keyName>
  <amiLaunchIndex>2</amiLaunchIndex>
  <InstanceType>m1.small</InstanceType>
  <launchTime>2007-08-07T11:51:50.000Z</launchTime>
</item>
</instancesSet>
</RunInstancesResponse>
```

## Related Operations

- [DescribeInstances](#)
- [TerminateInstances](#)
- [AuthorizeSecurityGroupIngress](#)
- [RevokeSecurityGroupIngress](#)
- [DescribeSecurityGroups](#)

## TerminateInstances

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

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

## Request Parameters

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

Name	Description	Required
<i>InstanceId.n</i>	One or more instance IDs returned. Type: String	Yes

## Response Elements

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

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

## Sample Request

```
https://ec2.amazonaws.com/  
?Action=TerminateInstances  
&InstanceId.1=i-2ea64347  
&InstanceId.2=i-21a64348  
&...auth parameters...
```

## Sample Response

```
<TerminateInstancesResponse xmlns="http://ec2.amazonaws.com/doc/2007-08-29">  
  <instancesSet>  
    <item>  
      <instanceId>i-28a64341</instanceId>  
      <shutdownState>  
        <code>32</code>  
        <name>shutting-down</name>  
      </shutdownState>  
      <previousState>  
        <code>16</code>  
        <name>running</name>  
      </previousState>  
    </item>  
    <item>  
      <instanceId>i-21a64348</instanceId>  
      <shutdownState>  
        <code>32</code>  
        <name>shutting-down</name>  
      </shutdownState>  
      <previousState>  
        <code>16</code>  
        <name>running</name>  
      </previousState>  
    </item>  
  </instancesSet>  
</TerminateInstancesResponse>
```

## Related Operations

- [DescribeInstances](#)

# Command Line Tools Reference

## Introduction

The Amazon EC2 command line tools provide a command line interface to the web service API. This section describes each tool and its command line arguments in detail.

Command line options and arguments are based on the GNU getopt conventions. Parameters are invoked using flags, which typically come in short and long form. In their short form, flags use a single character preceded by a dash. In their long form, flags use a more expressive name preceded by two dashes.



### Note

Some common options apply to all command line tools. These are described below and are not included in the description of the specific tools.

## Errors

Any service errors encountered by the command line tools are passed straight through from the API. For more information about these errors, see [API Error Codes](#).

## Common Options

Most command line tools described in this section accept the the following set of optional parameters:

Option	Description
<code>-U URL</code>	<code>URL</code> is the uniform resource locator of the Amazon EC2 web service entry point. Default: The <code>EC2_URL</code> environment variable, or <code>http://ec2.amazonaws.com</code> if the environment variable is not set. Example: <code>-U http://ec2.amazonaws.com</code>
<code>-K EC2-PRIVATE-KEY</code>	The private key to use when constructing requests to Amazon EC2.

Option	Description
	Default: The value of the <code>EC2_PRIVATE_KEY</code> environment variable. Example: <code>-k pk-HKZYKTAIG2ECMXIIBH3HXV4ZBZQ55CLO.pem</code>
<code>-C EC2-CERT</code>	The X.509 certificate to use when constructing requests to Amazon EC2. Default: The value of the <code>EC2_CERT</code> environment variable. Example: <code>-C cert-HKZYKTAIG2ECMXIIBH3HXV4ZBZQ55CLO.pem</code>
<code>-v</code>	Displays verbose output by showing the SOAP request and response on the command line. This is particularly useful if you are building tools to talk directly to our SOAP API.
<code>--show-empty-fields</code>	Shows empty columns as <code>(nil)</code> .
<code>--debug</code>	Prints internal debugging information. This is useful to assist us when troubleshooting problems.
<code>-?</code>	Displays help.
<code>-</code>	If <code>-</code> is specified as an argument to one of the parameters, a list of arguments are read from standard input. This is useful for piping the output of one command into the input of another. Example: <code>ec2-describe-instances   grep running   cut -f 2   ec2-terminate-instances -i -</code>

## AMI Tools

This section describes each tool used to create AMIs and its command line arguments in detail.

### ec2-bundle-image

#### Syntax

```
ec2-bundle-image -k private_key -c ec2_cert -u user_id -i image_path -r {i386 | x86_64} [-d destination] [-p ami_prefix] [-b, --batch]
```

#### Description

Create a bundled AMI from an operating system image created in a loopback file. For more information, see [Creating an AMI through a Loopback File](#).



#### Note

Scripts that require a copy of the public key from the launch key pair must obtain the key from the instance's metadata (not the key file in the ephemeral store) for instances bundled with the 2007-08-29 AMI tools and later. AMIs bundled before this release will continue to work normally.

## Options



### Note

This tool does not support the [common options](#)

Option	Description	Required
<code>-k, --privatekey private_key</code>	The path to the user's PEM-encoded RSA key file. Example: <code>-k pk-HKZYKTAIG2ECMXIYIBH3HXV4ZBZQ55CLO.pem</code>	Yes
<code>-c, --cert ec2_cert</code>	The user's PEM encoded RSA public key certificate file. Example: <code>-c cert-HKZYKTAIG2ECMXIYIBH3HXV4ZBZQ55CLO.pem</code>	Yes
<code>-u, --user user_id</code>	The user's AWS account number without dashes. Do not use the Access Key ID. Example: <code>-u 495219933132</code>	Yes
<code>-i, --image image_path</code>	The path to the image to bundle. Example: <code>-i /var/spool/my-image/version-2/debian.img</code>	Yes
<code>-r, --arch {i386   x86_64}</code>	Specifies 32-bit (m1.small) or 64-bit architecture (m1-large and m1-xlarge). Default: None Example: <code>-r x86_64</code>	Yes
<code>-d, --destination destination</code>	The directory in which to create the bundle. Default: The current directory Example: <code>-d /var/run/my-bundle</code>	No
<code>-p, --prefix ami_prefix</code>	The filename prefix for bundled AMI files. Default: image Example: <code>-p my-image-is-special</code>	No
<code>--help</code>	Display the help message. Example: <code>--help</code>	No
<code>--manual</code>	Display the manual entry. Example: <code>--manual</code>	No
<code>-b, --batch</code>	Runs without interaction and suppresses all warnings. Will attempt to automatically determine architecture. Example: <code>-b</code>	No

## Output

Status messages describing the stages and status of the bundling process.

## Example

This example creates a bundled AMI from an operating system image that was created in a loopback file:

```
$ ec2-bundle-image -k pk-HKZYKTAIG2ECMXIYIBH3HXV4ZBZQ55CLO.pem -c cert-  
HKZYKTAIG2ECMXIYIBH3HXV4ZBZQ55CLO.pem -u 495219933132 -i image.img -d bundled/  
-p fred -r x86_64  
Splitting bundled/fred.gz.crypt...  
Created fred.part.00  
Created fred.part.01  
Created fred.part.02  
Created fred.part.03  
Created fred.part.04  
Created fred.part.05  
Created fred.part.06  
Created fred.part.07  
Created fred.part.08  
Created fred.part.09  
Created fred.part.10  
Created fred.part.11  
Created fred.part.12  
Created fred.part.13  
Created fred.part.14  
Generating digests for each part...  
Digests generated.  
Creating bundle manifest...  
Bundle Image complete.
```

## Related Topics

- [ec2-bundle-vol](#)
- [ec2-unbundle](#)
- [ec2-upload-bundle](#)
- [ec2-download-bundle](#)
- [ec2-delete-bundle](#)

## ec2-bundle-vol

### Syntax

```
ec2-bundle-vol -k private_key -u user_id -c ec2_cert -r {i386 | x86_64} [-s  
size] [-d destination] [-e exclude_directory_1,exclude_directory_1,...] [-p  
ami_prefix] [-v volume] [--ec2cert ami_path] [--fstab fstab_path] [-b,  
--batch]
```

### Description

Creates a bundled AMI by compressing, encrypting and signing a snapshot of the local machine's root file system.



#### Note

Scripts that require a copy of the public key from the launch key pair must obtain the key



from the instance's metadata (not the key file in the ephemeral store) for instances bundled with the 2007-08-29 AMI tools and later. AMIs bundled before this release will continue to work normally.

## Options



### Note

This tool does not support the [common arguments](#)

Option	Description	Required
<code>-k, --privatekey private_key</code>	The path to the user's PEM-encoded RSA key file. Example: <code>-k pk-HKZYKTAIG2ECMXYIBH3HXV4ZBZQ55CLO.pem</code>	Yes
<code>-c, --cert ec2_cert</code>	The user's PEM encoded RSA public key certificate file. Example: <code>-c cert-HKZYKTAIG2ECMXYIBH3HXV4ZBZQ55CLO.pem</code>	Yes
<code>-u, --user user_id</code>	The user's AWS account number without dashes. Do not use the Access Key ID. Example: <code>-u 495219933132</code>	Yes
<code>-r, --arch {i386   x86_64}</code>	Specifies 32-bit (m1.small) or 64-bit architecture (m1-large and m1-xlarge). Example: <code>-r i386</code>	Yes
<code>-s, --size size</code>	The size, in MB (1024 * 1024 bytes), of the image file to create. The maximum size is 10240 MB. Default: 10240 Example: <code>-s 2048</code>	No
<code>-d, --destination destination</code>	The directory in which to create the bundle. Default: <code>/tmp</code> Example: <code>-d /var/run/my-bundle</code>	No
<code>-e, --exclude directory_1,directory_2,...</code>	A list of absolute directory paths to exclude from the bundle operation. This overrides the <code>--all</code> parameter. Example: <code>-e /tmp,/home/secret-data</code>	No
<code>-p, --prefix ami_prefix</code>	The filename prefix for bundled AMI files. Default: <code>image</code> Example: <code>-p my-image-is-special</code>	No
<code>-v, --volume volume</code>	The absolute path to the mounted volume from which to create the bundle. Default: The root directory ( <code>/</code> )	No

Option	Description	Required
	Example: <code>-v /mnt/my-customized-ami</code>	
<code>-a, --all</code>	Bundle all directories, including those on remotely mounted filesystems. Example: <code>-a</code>	No
<code>--ec2cert ami_path</code>	The path to the Amazon EC2 X509 public key certificate. Default: <code>/etc/aes/amiutil/cert-ec2.pem</code> Example: <code>--ec2cert /etc/aes/amiutil/cert-ec2.pem</code>	No
<code>--help</code>	Display the help message. Example: <code>--help</code>	No
<code>--manual</code>	Display the manual entry. Example: <code>--manual</code>	No
<code>--fstab fstab_path</code>	The path to the fstab to bundle into the image. If this is not specified, Amazon EC2 will provide its own fstab. Example: <code>--fstab /etc/fstab</code>	No
<code>-b, --batch</code>	Runs without interaction and suppresses all warnings. Will attempt to automatically determine architecture. Example: <code>-b</code>	No

## Output

Status messages describing the stages and status of the bundling

## Example

This example creates a bundled AMI by compressing, encrypting and signing a snapshot of the local machine's root file system.

```
$ ec2-bundle-vol -d /mnt -k pk-HKZYKTAIG2ECMXIYBH3HXV4ZBZQ55CLO.pem -c cert-
HKZYKTAIG2ECMXIYBH3HXV4ZBZQ55CLO.pem -u 495219933132 -r x86_64
Copying / into the image file /mnt/image.img...
Excluding:
  sys
  dev/shm
  proc
  dev/pts
  proc/sys/fs/binfmt_misc
  dev
  media
  mnt
  proc
  sys
  tmp/image.img
  mnt/img-mnt
1+0 records in
1+0 records out
mke2fs 1.38 (30-Jun-2005)
```

```
warning: 256 blocks unused.  
  
Splitting /mnt/image.gz.crypt...  
Created image.part.00  
Created image.part.01  
Created image.part.02  
Created image.part.03  
...  
Created image.part.22  
Created image.part.23  
Generating digests for each part...  
Digests generated.  
Creating bundle manifest...  
Bundle Volume complete.
```

## Related Topics

- [ec2-bundle-image](#)
- [ec2-unbundle](#)
- [ec2-upload-bundle](#)
- [ec2-download-bundle](#)
- [ec2-delete-bundle](#)

## ec2-delete-bundle

### Syntax

```
ec2-delete-bundle -b s3_bucket -a access_key_id -s secret_key [-m  
manifest_path] [-p ami_prefix] [--url url] [--retry] [-y]
```

### Description

Deletes the specified bundle from Amazon S3 storage.

### Options



#### Note

This tool does not support the [common arguments](#)

Option	Description	Required
-b, --bucket <i>s3_bucket</i>	The name of the Amazon S3 bucket containing the bundled AMI Example: -b aes-cracker-ami-bucket	Yes
-a, --access-key <i>access_key_id</i>	The AWS access key ID. Example: -a 10QMXFEV71ZS32XQFTR2	Yes
-s, --secret-key <i>secret_key</i>	The AWS secret access key. Example: -s DMADSSfPfdaDjbK+RRUhS/aDrjsiZadgAUm8gRU2	Yes

Option	Description	Required
<code>-m, --manifest manifest_path</code>	The path to the unencrypted manifest file. Example: <code>-m /var/spool/my-first-bundle/Manifest</code>	No
<code>-p, --prefix ami_prefix</code>	The bundled AMI filename prefix. Example: <code>-p eos-</code>	No
<code>--url url</code>	The Amazon S3 service URL. Default: <code>https://s3.amazonaws.com</code> Example: <code>--url https://s3.amazonaws.ie</code>	No
<code>--retry</code>	Automatically retries failed uploads. Use with caution. Example: <code>--retry</code>	No
<code>-y, --yes</code>	Automatically assumes the answer to all prompts is 'yes'. Example: <code>-y</code>	No
<code>--help</code>	Display the help message. Example: <code>--help</code>	No
<code>--manual</code>	Display the manual entry. Example: <code>--manual</code>	No

## Output

Amazon EC2 displays status messages indicating the stages and status of the delete process.

## Example

This example deletes a bundle from Amazon S3:

```
$ ec2-delete-bundle -b my-s3-bucket -a 10QMXFEV71ZS32XQFTR2 -s DMADSSfPfdad
jBk+RRUhs/aDrjsiZadgAUm8gRU2 -p fred
Deleting files:
my-s3-bucket/fred.manifest.xml
my-s3-bucket/fred.part.00
my-s3-bucket/fred.part.01
my-s3-bucket/fred.part.02
my-s3-bucket/fred.part.03
my-s3-bucket/fred.part.04
my-s3-bucket/fred.part.05
my-s3-bucket/fred.part.06
Continue? [y/n]
y
Deleted my-s3-bucket/fred.manifest.xml
Deleted my-s3-bucket/fred.part.00
Deleted my-s3-bucket/fred.part.01
Deleted my-s3-bucket/fred.part.02
Deleted my-s3-bucket/fred.part.03
Deleted my-s3-bucket/fred.part.04
Deleted my-s3-bucket/fred.part.05
Deleted my-s3-bucket/fred.part.06
ec2-delete-bundle complete.
```

## Related Topics

- [ec2-bundle-image](#)
- [ec2-bundle-vol](#)
- [ec2-unbundle](#)
- [ec2-upload-bundle](#)
- [ec2-download-bundle](#)

## ec2-download-bundle

### Syntax

```
ec2-download-bundle -b s3_bucket -m manifest -a access_key_id -s secret_key
-k private_key [-p ami_prefix] [-d directory] [--url url]
```

### Description

Download the specified bundles from S3 storage.

### Options



#### Note

This tool does not support the [common arguments](#)

Option	Description	Required
<i>b</i> , --bucket <i>s3_bucket</i>	The name of the Amazon S3 bucket where the bundle is located. Example: <i>-b aes-cracked</i>	Yes
<i>-m</i> , --manifest <i>manifest</i>	The manifest path and filename. Example: <i>-m /var/spool/my-first-bundle/Manifest</i>	Yes
<i>-a</i> , --access-key <i>access_key_id</i>	Your AWS access key ID. Example: <i>-a 10QMXFEV71ZS32XQFTR2</i>	Yes
<i>-s</i> , --secret-key <i>secret_key</i>	Your AWS secret access key. Example: <i>-s DMADSSfPfdaDjbK+RRUhS/aDrjsiZadgAUm8gRU2</i>	Yes
<i>-k</i> , --privatekey <i>private_key</i>	The private key used to decrypt the manifest. Example: <i>-k pk-HKZYKTAIG2ECMXIYBH3HXV4ZBZQ55CLO.pem</i>	Yes
<i>-p</i> , --prefix <i>ami_prefix</i>	The filename prefix for the bundled AMI files. Default: <i>image</i> Example: <i>-p my-image</i>	No

Option	Description	Required
<code>-d, --directory <i>directory</i></code>	The directory where the downloaded bundles are saved. The directory must exist. Default: The current working directory Example: <code>-d /tmp/my-downloaded-bundle</code>	No
<code>--url <i>url</i></code>	The S3 service URL. Default: <code>https://s3.amazonaws.com</code> Example: <code>--url https://s3.amazonaws.ie</code>	No
<code>--help</code>	Display the help message. Example: <code>--help</code>	No

## Output

Status messages indicating the various stages of the download process are displayed.

## Example

This example downloads creates the `bundled` directory and downloads the bundle from the `my-s3-bucket` Amazon S3 bucket.

```
$ mkdir bundled
$ ec2-download-bundle -b my-s3-bucket -m fred.manifest.xml -a
10QMXFEV71ZS32XQFTR2 -s DMADSSfpfdaDjbK+RRUhs/aDrjsiZadgAUm8gRU2 -k pk-
HKZYKTAIG2ECMXIYBH3HXV4ZBZQ55CLO.pem -d bundled
downloading manifest https://s3.amazonaws.com/my-s3-bucket/image.manifest.xml
to bundled/image.manifest.xml ...
downloading part https://s3.amazonaws.com/my-s3-bucket/image.part.00 to
bundled/image.part.00 ...
Downloaded image.part.00 from https://s3.amazonaws.com/my-s3-bucket.
downloading part https://s3.amazonaws.com/my-s3-bucket/image.part.01 to
bundled/image.part.01 ...
Downloaded image.part.01 from https://s3.amazonaws.com/my-s3-bucket.
downloading part https://s3.amazonaws.com/my-s3-bucket/image.part.02 to
bundled/image.part.02 ...
Downloaded image.part.02 from https://s3.amazonaws.com/my-s3-bucket.
downloading part https://s3.amazonaws.com/my-s3-bucket/image.part.03 to
bundled/image.part.03 ...
Downloaded image.part.03 from https://s3.amazonaws.com/my-s3-bucket.
downloading part https://s3.amazonaws.com/my-s3-bucket/image.part.04 to
bundled/image.part.04 ...
Downloaded image.part.04 from https://s3.amazonaws.com/my-s3-bucket.
downloading part https://s3.amazonaws.com/my-s3-bucket/image.part.05 to
bundled/image.part.05 ...
Downloaded image.part.05 from https://s3.amazonaws.com/my-s3-bucket.
downloading part https://s3.amazonaws.com/my-s3-bucket/image.part.06 to
bundled/image.part.06 ...
Downloaded image.part.06 from https://s3.amazonaws.com/my-s3-bucket.
Download Bundle complete.
```

## Related Topics

- [ec2-bundle-image](#)
- [ec2-bundle-vol](#)
- [ec2-unbundle](#)

- [ec2-upload-bundle](#)
- [ec2-delete-bundle](#)

## ec2-unbundle

### Syntax

```
ec2-unbundle -m manifest -k private_key [-d destination_directory] [-s source_directory]
```

### Description

Recreates the AMI from the bundled AMI parts.

### Options



#### Note

This tool does not support the [common arguments](#)

Option	Description	Required
-m, --manifest <i>manifest</i>	The path to the unencrypted AMI manifest file. Example: -m /var/spool/my-first-bundle/Manifest	Yes
-k, --privatekey <i>private_key</i>	The path to your PEM-encoded RSA key file. Example: -k \$HOME/pk-234242DEADCAFE.pem	Yes
-d, --destination <i>destination_directory</i>	The directory in which to unbundle the AMI. The destination directory must exist. Default: The current directory Example: -d /tmp/my-image	No
-s, --source <i>source_directory</i>	The directory containing the bundled AMI parts. Default: The current directory Example: -s /tmp/my-bundled-image	No
--help	Display the help message. Example: --help	No

### Example

This example unbundles the AMI specified in the `fred.manifest.xml` file:

```
$ mkdir unbundled
$ ec2-unbundle -m fred.manifest.xml -s bundled -d unbundled
cat bundled/fred.part.00 bundled/fred.part.01 bundled/fred.part.02 bundled/
fred.part.03 bundled/fred.part.04 bundled/fred.part.05 bundled/fred.part.06
bundled/fred.part.07 bundled/fred.part.08 bundled/fred.part.09 bundled/
fred.part.10 bundled/fred.part.11 bundled/fred.part.12 bundled/fred.part.13
```

```
bundled/fred.part.14 | openssl enc -d -aes-128-cbc -K
a8f8e9586b7fd3df893b237f88e351a9 -iv 121febdf64b0322cd4ffda03aalab535 | gun
zip > unbundled/fred.img
Unbundle complete.
$ ls -l unbundled
total 1025008
-rw-r--r-- 1 root root 1048578048 Aug 25 23:46 fred.img
```

## Output

Status messages indicating the various stages of the unbundling process are displayed.

## Related Topics

- [ec2-bundle-image](#)
- [ec2-bundle-vol](#)
- [ec2-upload-bundle](#)
- [ec2-download-bundle](#)
- [ec2-delete-bundle](#)

## ec2-upload-bundle

### Syntax

```
ec2-upload-bundle -b s3_bucket -m manifest -a access_key_id -s secret_key
[--acl acl] [--ec2cert certificate] [-d directory] [--part part] [--url url]
[--retry] [--skipmanifest]
```

### Description

Upload a bundled AMI to Amazon S3 storage.

### Options



#### Note

This tool does not support the [common options](#)

Option	Description	Required
<code>-b, --bucket <i>s3_bucket</i></code>	The name of the Amazon S3 bucket in which to store the bundle. If the bucket doesn't exist it will be created (if the bucket name is available). Example: <code>-b aes-cracker-ami</code>	Yes
<code>-m, --manifest <i>manifest</i></code>	The path to the manifest file. The manifest file is created during the bundling process and can be found in the directory containing the bundle. Example: <code>-m /var/spool/my-first-bundle/Manifest</code>	Yes
<code>-a, --access-key</code>	Your AWS access key ID.	Yes



Option	Description	Required
<code>access_key_id</code>	Example: <code>-a 10QMXFEV71ZS32XQFTR2</code>	
<code>-s, --secret-key secret_key</code>	Your AWS secret access key. Example: <code>-s DMADSSfPfdaDjbK+RRUhS/aDrjsiZadgAUm8gRU2</code>	Yes
<code>--acl acl</code>	The access control list policy of the bundled image. Valid Values: <code>public-read   aws-exec-read</code> Default: <code>aws-exec-read</code> Example: <code>--acl public-read</code>	No
<code>--ec2cert certificate</code>	The path to the Amazon EC2 X509 public key certificate. Default: <code>/etc/aes/amiutil/cert-ec2.pem</code> Example: <code>--ec2cert /etc/aes/amiutil/cert-ec2.pem</code>	No
<code>-d, --directory directory</code>	The directory containing the bundled AMI parts. Default: The directory containing the manifest file (see the <code>-m</code> option). Example: <code>-d /var/run/my-bundle</code>	No
<code>--part part</code>	Starts uploading the specified part and all subsequent parts. Example: <code>--part</code>	No
<code>--url url</code>	The S3 service URL. Default: <code>https://s3.amazonaws.com</code> Example: <code>--url https://s3.amazonaws.ie</code>	No
<code>--retry</code>	Automatically retries failed uploads. Use with caution. Example: <code>--retry</code>	No
<code>--skipmanifest</code>	Does not upload the manifest. Example: <code>--skipmanifest</code>	No
<code>--help</code>	Display the help message. Example: <code>--help</code>	No
<code>--manual</code>	Display the manual entry. Example: <code>--manual</code>	No

## Output

Amazon EC2 displays status messages that indicate the stages and status of the upload process.

## Example

This example uploads the bundle specified by the `bundled/fred.manifest.xml` manifest:

```
$ ec2-upload-bundle -b my-s3-bucket -m bundled/fred.manifest.xml -a
10QMXFEV71ZS32XQFTR2 -s DMADSSfPfdADjbK+RRUhS/aDrjsiZadgAUm8gRU2 -d bundled
Encrypting bundle manifest...
Completed encryption.
Uploading encrypted manifest...
Uploaded encrypted manifest to ht
tp://s3.amazonaws.com:80/alpowell-images/fred.manifest.xml.
Uploading bundled AMI parts to http://s3.amazonaws.com:80/alpowell-images...
Uploaded fred.part.00 to ht
tp://s3.amazonaws.com:80/alpowell-images/fred.part.00.
Uploaded fred.part.01 to ht
tp://s3.amazonaws.com:80/alpowell-images/fred.part.01.
Uploaded fred.part.02 to ht
tp://s3.amazonaws.com:80/alpowell-images/fred.part.02.
Uploaded fred.part.03 to ht
tp://s3.amazonaws.com:80/alpowell-images/fred.part.03.
Uploaded fred.part.04 to ht
tp://s3.amazonaws.com:80/alpowell-images/fred.part.04.
Uploaded fred.part.05 to ht
tp://s3.amazonaws.com:80/alpowell-images/fred.part.05.
Uploaded fred.part.06 to ht
tp://s3.amazonaws.com:80/alpowell-images/fred.part.06.
Uploaded fred.part.07 to ht
tp://s3.amazonaws.com:80/alpowell-images/fred.part.07.
Uploaded fred.part.08 to ht
tp://s3.amazonaws.com:80/alpowell-images/fred.part.08.
Uploaded fred.part.09 to ht
tp://s3.amazonaws.com:80/alpowell-images/fred.part.09.
Uploaded fred.part.10 to ht
tp://s3.amazonaws.com:80/alpowell-images/fred.part.10.
Uploaded fred.part.11 to ht
tp://s3.amazonaws.com:80/alpowell-images/fred.part.11.
Uploaded fred.part.12 to ht
tp://s3.amazonaws.com:80/alpowell-images/fred.part.12.
Uploaded fred.part.13 to ht
tp://s3.amazonaws.com:80/alpowell-images/fred.part.13.
Uploaded fred.part.14 to ht
tp://s3.amazonaws.com:80/alpowell-images/fred.part.14.
Upload Bundle complete.
```

## Related Topics

- [ec2-bundle-image](#)
- [ec2-bundle-vol](#)
- [ec2-unbundle](#)
- [ec2-download-bundle](#)
- [ec2-delete-bundle](#)

## API Tools

This section describes each API tool and its command line arguments in detail.

## List of Operations by Function

### Images

- [ec2-register](#)

- [ec2-deregister](#)
- [ec2-describe-images](#)

### Instances

- [ec2-run-instances](#)
- [ec2-describe-instances](#)
- [ec2-terminate-instances](#)
- [ec2-confirm-product-instance](#)

### Key Pairs

- [ec2-add-keypair](#)
- [ec2-describe-keypairs](#)
- [ec2-delete-keypair](#)
- [ec2-fingerprint-key](#)

### Image Attributes

- [ec2-modify-image-attribute](#)
- [ec2-describe-image-attribute](#)
- [ec2-reset-image-attribute](#)

### Security Groups

- [ec2-add-group](#)
- [ec2-delete-group](#)
- [ec2-describe-groups](#)
- [ec2-authorize](#)
- [ec2-revoke](#)

## ec2-add-group

### Syntax

```
ec2-add-group group -d description
```

### Description

Creates a new security group. Group names must be unique per account.

### Options

Option	Description	Required
<i>group</i>	Name of the security group.	Yes

Option	Description	Required
	Example: <code>webservers</code>	
<code>-d <i>description</i></code>	Description of the group. This is informational only. If the description contains spaces, you must enclose it in single quotes (''). Example: <code>-d 'Web servers'</code>	Yes

## Output

Amazon EC2 returns a table that contains the following information:

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

Amazon EC2 displays errors on stderr.

## Example

This example creates the `websrv` security group.

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

## Related Topics

- [CreateSecurityGroup](#)
- [ec2-describe-groups](#)
- [ec2-delete-group](#)
- [ec2-authorize](#)
- [ec2-revoke](#)

## ec2-add-keypair

### Syntax

```
ec2-add-keypair key
```

### Description

Creates a new 2048 bit RSA key pair with the specified name. The public key is stored by Amazon EC2 and the private key is displayed on the console. The private key is returned as an unencrypted PEM encoded PKCS#8 private key. If a key with the specified name already exists, Amazon EC2 returns an error.

### Options

Option	Description	Required
<i>key</i>	Name of the key pair. Example: mysecretkey	Yes

## Output

Amazon EC2 returns a table that contains the following information:

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

Amazon EC2 displays errors on stderr.

## Example

This example creates a key pair named gsg-keypair:

```
$ ec2-add-keypair gsg-keypair
KEYPAIR gsg-keypair
1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:f1:6f
-----BEGIN RSA PRIVATE KEY-----
MIIEoQIBAAKCAQBULFg5ujHrtm1jnutSuoO8Xe56LlT+HM8v/xkaa39EstM3/aFxTHgElQiJLChp
HungXQ29VTc8rc1bW0lkdi23OH5eqkMHGhvEwqa0HWASUM1l4o3o/IX+0f2UcPoKCOVUR+jx7lSg
5AU52EQfanIn3ZQ8lFW7Edp5a3q4DhjGluKToHVbicL5E+g45zfB95wIyywWZfeW/UUF3LpGZyq/
ebiU1q1qTbHkLbCC2r7RTn8vpQWp47BGVYgtGSBMpTRP5hnbzquqj3itkiLHjU39S2sJCJ0TrJx5
i8BygR4s3mHKBj8l+ePQxG1kGbF6R4yg6sECmXn17MRQVXODNHZbAgMBAAECggEAYltsiUsIwDl5
91CXirkYGuVfLyLflXenxfi50mDFms/mumTqloHO7tr0oriHDR5K7wMcY/YY5YkcXNo7mvUVDlpM
ZNUJs7rw9gZRTf7LylaJ58kOcyajw8TsC4e4LPbFaHwS1d6K8rXh64o6WgW4SrsB6ICmr1kGQI7
3wcfgt5ecIu4Tzf00E9IHjn+2eRlSrjBdeORi7KiUNC/pAG23I6MdDOFEQRcCSigCj+4/mciFUSA
SWS4dMbrpb9FNSIcf9dcLxVM7/6KxgJNfZc9XWzUw77Jg8x92Zd0fVhHoux5IZC+UvSKWB4dyfcI
tE8C3p9bbU9VGyY5vLCAiIb4qQKBgQDLiO24GXrIkswF32YtBBMuVgLCwU9h9HlO9mKAc2m8Cm1
jUE5IpzRjTcdc9I2qiIMUTwtgnw42auSCzbUeYmURPtDqyQ7p6A jMuJp9EPemcSVOK9vXYL0Ptco
xW9MC0dtV6iPkCN7gOqiZXPRKaFbWADp16p8UAIvS/a5XXk5jwKBgQCKkphi2EiShluRkhxljyWC
iDCiK6JBRsMvplbc0v5dKwP5alo1fmdR5PJaV2qvZSj5CYNpMay1/EDNTY5OSIJU+0KFmQbyhsbm
rdLNLDL4+Tcnt7c62/aH01ohYaf/VCBRhtLlBfgGoQc7+sAc8vmKkesnF7CqCEKdyF/dhrxYdQKB
gC0iZzzNAapayz1+JcVTwwEid6j9JqNXbBc+Z2YwMi+T0Fv/P/hwkX/ypeOXnIUcw0Ih/YtGBVAC
DQbsz7LcY1HqXiHKYNWNvXgwwO+oiChjxvEkSdsTTIfnK4VSCvU9BxDbQHjdiNDJbL6oar92UN7V
rBYvChJZF7LvUH4YmVpHAoGAbZ2X7XvoeEO+uZ58/BGKOIGHByHBDiXtzMhdJr15HTYjxK70gTZm
gK+8zp4L9IbvLGDMJO8vft32XPEWuvI8twCzFH+CswLQADZMZKSsBasOZ/h1FwhdMgCMcY+Qlzd4
JZKjTSu3i7vhvx6RzdSedXEMNTZWN4qlIx3kR5aHcukCgYA9T+ZrvmlF0seQPbLknn7EqhXIjBaT
P8TTvW/6bdPi23ExzxZn7KodrflYRphlLHMPaONv/x2xALIf91UB+v5ohyloDoasL0gijlhouRe
2ERKKdwz0ZL9SWq6VTdhr/5G994CK72fy5WhyERbdjUiDHaK3M849Jjuf8cSrvSb4g==
-----END RSA PRIVATE KEY-----
```

## Related Topics

- [CreateKeypair](#)
- [ec2-describe-keypairs](#)
- [ec2-delete-keypair](#)

## ec2-authorize

## Syntax

```
ec2-authorize group [-P protocol] (-p port_range | -t icmp_type_code) [-u source_group_user ...] [-o source_group ...] [-s source_subnet ...]
```

## Description

Adds a rule to the specified security group. If no source host, group or subnet is provided, requests from any source address will be honored.

## Options

Option	Description	Required
<i>group</i>	The group to which this rule will apply. Example: webserver	Yes
<i>-P protocol</i>	The protocol to allow. Condition: Applies when specifying a CIDR subnet as the source. Valid Values: tcp   udp   icmp Example: -P tcp	Yes
<i>-p port_range</i>	For the TCP or UDP protocols, this specifies the range of ports to allow. You can specify a single integer or a range (min-max). Condition: Applies when specifying a CIDR subnet as the source. Example: -p 80	Yes
<i>-t icmp_type_code</i>	For the ICMP protocol, the ICMP type and code must be specified. This must be specified in the format type:code where both are integers. Type, code, or both can be specified as -1, which is a wildcard. Condition: Applies when specifying a CIDR subnet as the source. Example: -t 2:5	Yes
<i>-u source_group_user</i>	The owner of a group specified using -o. If this is not specified, all groups will refer to the current user. If specified more than once, there must be exactly one -u per -o and each user will be mapped to the corresponding group. Example: -u 495219933132	No
<i>-o source_group</i>	The network source from which traffic will be authorized specified as a security Group. See the description of the -u option for group owner information. Example: -o headoffice	No
<i>-s source_subnet</i>	The network source from which traffic is to be authorized specified as a CIDR subnet range. Example: -s 205.192.8.45/24	No

Option	Description	Required

## Output

Amazon EC2 returns a table that contains the following information:

- Output type identifier ("GROUP", "PERMISSION").
- Group name. Currently, this will report an empty string.
- Type of rule. Currently, only ALLOW rules are supported.
- Protocol to allow.
- Start of port range.
- End of port range.
- FROM
- Source.

Amazon EC2 displays errors on stderr.

## Example

This example grants TCP port 80 access from the 205.192.0.0/16 address range to the `webserv` security group:

```
$ ec2-authorize webserv -P tcp -p 80 -s 205.192.0.0/16
GROUP webserv ""
PERMISSION webserv ALLOWS tcp 80 80 FROM CIDR 205.192.0.0/16
```

## Related Topics

- [AuthorizeSecurityGroupIngress](#)
- [ec2-add-group](#)
- [ec2-describe-groups](#)
- [ec2-delete-group](#)
- [ec2-revoke](#)

## ec2-confirm-product-instance

### Syntax

```
ec2-confirm-product-instance product_code -i instance_id
```

### Description

Returns a boolean indicating whether the specified product code is attached to the specified instance. If it is attached, It returns true. Otherwise, it returns false.

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

## Options

Option	Description	Required
<i>instance_id</i>	Instance identifier that was generated when the instance launched. Example: i-10a64379	Yes
<i>product_code</i>	The product code. Example: 774F4FF8	Yes

## Output

Amazon EC2 returns a table that contains the following information:

- Product code.
- Instance ID.
- Boolean indicating if the product code is attached to the instance.
- The instance owner's account ID (if the product code is attached).

Amazon EC2 displays errors on stderr.

## Example

This example confirms whether the 774F4FF8 product code is attached to the i-10a64379 instance:

```
$ ec2-confirm-product-instance 774F4FF8 -i i-10a64379
774F4FF8 i-10a64379 true
```

## Related Topics

- [DescribeInstances](#)
- [ec2-modify-image-attribute](#)

## ec2-delete-group

### Syntax

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

### Description

Deletes the specified security group.



#### Note

If you attempt to delete a security group that contains instances, a fault is returned. If you attempt to delete a security group that is referenced by another security group, a fault is returned. For example, if security group B has a rule that allows access from



security group A, security group A cannot be deleted until the allow rule is removed.

## Options

Option	Description	Required
<i>group</i>	Name of the security group. Example: webservers	Yes

## Output

Amazon EC2 returns a table that contains the following information:

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

Amazon EC2 displays errors on stderr.

## Example

This command deletes the *websrv* security group:

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

## Related Topics

- [DeleteSecurityGroup](#)
- [ec2-add-group](#)
- [ec2-describe-groups](#)
- [ec2-authorize](#)
- [ec2-revoke](#)

## ec2-delete-keypair

### Syntax

```
ec2-delete-keypair key_pair
```

### Description

Deletes the specified key pair, by removing the public key from Amazon EC2

### Options

Option	Description	Required
<i>key_pair</i>	Name of the key pair.	Yes

Option	Description	Required
	Example: <code>primary_keypair</code>	

## Output

Amazon EC2 returns a table that contains the following information:

- Output type identifier ("KEYPAIR").
- Name of the deleted key pair.
- Private key fingerprint.

Amazon EC2 displays errors on `stderr`.

## Example

This example deletes the *gsg-keypair* key pair:

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

## Related Topics

- [DeleteKeypair](#)
- [ec2-add-keypair](#)
- [ec2-describe-keypairs](#)

# ec2-deregister

## Syntax

```
ec2-deregister ami_id
```

## Description

Deregisters the specified AMI. Once deregistered, the AMI cannot be used to launch new instances.



### Note

This command does not delete the AMI from Amazon S3.

## Options

Option	Description	Required
<i>ami_id</i>	AMI identifier. Example: <code>ami-4fa54026</code>	Yes

Option	Description	Required

## Output

Amazon EC2 returns a table that contains the following information:

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

Amazon EC2 displays errors on stderr.

## Example

This example deregisters the *ami-4fa54026* AMI:

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

## Related Topics

- [DeregisterImage](#)
- [ec2-register](#)
- [ec2-describe-images](#)

# ec2-describe-groups

## Syntax

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

## Description

Describes the current state of each specified security group. If no security groups are explicitly listed, Amazon EC2 displays all security groups owned by the current user.

## Options

Option	Description	Required
<i>group</i>	Name of the security group. Example: webserver	Yes

## Output

Amazon EC2 returns a table that contains the following information:

- Output type identifier ("GROUP", "PERMISSION").

- User ID of security group owner.
- Security group name.
- Description of the security group.
- Firewall rule.

Amazon EC2 displays errors on stderr.

## Example

This example displays the state of the `webserv` security group:

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

## Related Topics

- [DescribeSecurityGroups](#)
- [ec2-add-group](#)
- [ec2-delete-group](#)
- [ec2-authorize](#)
- [ec2-revoke](#)

# ec2-describe-image-attribute

## Syntax

```
ec2-describe-image-attribute ami_id ( -l | -p )
```

## Description

Describes an attribute for the specified AMI.

## Options

Option	Description	Required
<i>ami_id</i>	AMI identifier. Example: <code>ami-4fa54026</code>	Yes
<i>-l</i>	Describes the <code>launchPermission</code> attribute. Example: <code>-l</code>	Choice
<i>-p</i>	Describes the <code>productCodes</code> attribute. Example: <code>-p</code>	Choice

## Output

Amazon EC2 returns a table that contains the following information:

- Attribute type identifier
- ID of the AMI.
- Attribute value type or attribute list item value type.
- Attribute or attribute list item value.

Amazon EC2 displays errors on stderr.

## Examples

This example lists the launch permissions for the ami-2bb65342 AMI:

```
$ ec2-describe-image-attribute ami-2bb65342 -l  
launchPermission ami-2bb65342 group all  
launchPermission ami-2bb65342 userId 495219933132
```

This example lists the product code for the ami-2bb65342 AMI:

```
$ ec2-describe-image-attribute ami-2bb65342 -p  
productCodes ami-2bb65342 productCode 774F4FF8
```

## Related Topics

- [DescribeImageAttribute](#)
- [ec2-modify-image-attribute](#)
- [ec2-reset-image-attribute](#)
- [Sharing AMIs](#)

# ec2-describe-images

## Syntax

```
ec2-describe-images[ami_id ...] [-a] [-o owner ...] [-x user_id]
```

## Description

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

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

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

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

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

have explicit permissions, or *all* for public AMIs.



#### Note

If you do not specify any optional parameters, Amazon EC2 returns images you own or images for which you have explicit access. Public images are not returned.

## Options

Option	Description	Required
<code>-a</code>	Returns AMIs that the user owns and for which the user has execution permissions. Example: <code>-a</code>	No
<code>-o owner</code>	Returns AMIs owned by the specified owner. Multiple owners can be specified. Owners are specified with AWS user account ID, without dashes. The IDs <code>amazon</code> , <code>self</code> , and <code>explicit</code> can be used to include AMIs owned by Amazon, AMIs owned by the user, and AMIs for which the user has explicit launch permissions, respectively. Example: <code>-o 495219933132</code>	No
<code>-x user_id</code>	Returns AMIs for which the specified user has explicit launch permissions. The user ID can be a user's account ID, <code>self</code> to return AMIs for which the sender of the request has explicit launch permissions, or <code>all</code> to return AMIs with public launch permissions. Example: <code>-x self</code>	No

## Output

Amazon EC2 returns a table that contains the following information:

- A record type identifier ("IMAGE")
- Image identifier
- Manifest location
- User identifier of the user that registered the image
- Image status
- Image visibility (`public` or `private`)
- Product codes, if any, that are attached to the instance

Amazon EC2 displays errors on `stderr`.

## Example

This example describes the `ami-78a54011` AMI:

```
$ ec2-describe-images ami-78a54011
```

```
IMAGE ami-78a54011 powerdns/image.manifest.xml 495219933132 available private
774F4FF8
```

This example describes the Amazon 64-bit AMI::

```
$ ec2-describe-images -o amazon | grep x86_64
IMAGE    ami-78a54034
ec2-public-images/fedora-core6-base-x86_64.manifest.xml
amazon   available   public
```

## Related Topics

- [DescribeImages](#)
- [ec2-register](#)
- [ec2-deregister](#)

## ec2-describe-instances

### Syntax

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

### Description

Describes the current state of the specified instance(s). If you do not specify instances, all your instances are included in the output.

### Options

Option	Description	Required
<i>instance_id</i>	Instance identifier that was generated when the instance launched. Example: <code>r-15a4417c</code>	No

### Output

Amazon EC2 returns a table that contains the following information:

- Output type identifier ("RESERVATION", "INSTANCE")
- Instance ID for each running instance.
- AMI ID of the image on which the instance is based.
- Public DNS name associated with the instance. This is only present for instances in the `running` state.
- Private DNS name associated with the instance. This is only present for instances in the `running` state, but is not valid for instances using legacy direct addressing.
- Instance state.
- Key name. If a key was associated with the instance at launch, its name will appear.
- AMI launch index. For more information, see [Instance Metadata](#).
- Product codes attached to the instance.

- Instance type. The type of the instance (e.g., m1.small, m1.large, m1.xlarge).
- Instance launch time. The time the instance launched.

Amazon EC2 displays errors on stderr.

## Example

This example describes the current state of the instances (currently one) owned by this user:

```
$ ec2-describe-instances
RESERVATION r-15a4417c 495219933132
INSTANCE i-3ea74257 ami-6ba54002 ec2-72-44-33-4.compute-1.amazonaws.com domU-12-31-33-00-00-01.compute-1.internal running 0 774F4FF8 m1.small
2007-07-11T16:40:44+0000
INSTANCE i-31a74258 ami-6ba54002 ec2-72-44-34-23.compute-1.amazonaws.com domU-12-31-33-00-00-02.compute-1.internal running 1 m1.small
2007-07-11T16:40:44+0000
```

## Related Topics

- [DescribeInstances](#)
- [ec2-run-instances](#)
- [ec2-terminate-instances](#)

# ec2-describe-keypairs

## Syntax

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

## Description

Describes the current state of each specified key. If no keys are specified, all keys owned by the current user are included in the output.

## Options

Option	Description	Required
<i>key_id</i>	Names of one or more keys. Example: gsg-keypair	No

## Output

Amazon EC2 returns a table that contains the following information:

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



Amazon EC2 displays errors on stderr.

## Example

This example describes the state of the current keys:

```
$ ec2-describe-keypairs gsg-keypair
KEYPAIR gsg-keypair
1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:f1:6f
```

## Related Topics

- [DescribeKeypairs](#)
- [ec2-add-keypair](#)
- [ec2-delete-keypair](#)

# ec2-fingerprint-key

## Syntax

```
ec2-fingerprint-key keyfile
```

## Description

Computes and displays the fingerprint for a private key produced by Amazon EC2. .

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

## Options

Option	Description	Required
<i>keyfile</i>	The path to a file containing an unencrypted PEM-encoded PKCS#8 private key. Example: <code>mykey.pem</code>	Yes

## Output

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

Amazon EC2 displays errors on stderr.

## Example

This example computes and displays the fingerprint for the `mykey.pem` private key:

```
$ ec2-fingerprint-key mykey.pem
1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:f1:6f
```

## Related Topics

- [ec2-add-keypair](#)
- [ec2-describe-keypairs](#)

## ec2-get-console-output

### Syntax

```
ec2-get-console-output instance_id [-r]
```

### Description

Retrieves the console output for specified instance, if available, and displays it to `stdout`.

### Options

Option	Description	Required
<i>instance_id</i>	Instance identifier generated when the instance lauched. Example: i-10a64379	Yes
-r	Raw output. Do not escape the output to facilitate reading.	No

### Output

- A timestamp indicating the time of the last update.
- The instance console output. By default the `^ESC` character is escaped and duplicate new-lines are removed to facilitate reading.

Amazon EC2 displays errors on `stderr`.

### Example

This example retrieves the console output for the i-10a64379 instance

```
$ ec2-get-console-output i-10a64379
2007-01-03 12:00:00
Linux version 2.6.16-xenU (builder@patchbat.amazonsa) (gcc version 4.0.1
20050727 (Red Hat 4.0.1-5)) #1 SMP Thu Oct 26 08:41:26 SAST 2006
BIOS-provided physical RAM map:
Xen: 0000000000000000 - 000000006a400000 (usable)
980MB HIGHMEM available.
727MB LOWMEM available.
NX (Execute Disable) protection: active
IRQ lockup detection disabled
Built 1 zonelists
Kernel command line: root=/dev/sda1 ro 4
Enabling fast FPU save and restore... done.
```

## ec2-modify-image-attribute

## Syntax



```
ec2-modify-image-attribute ami_id -l (-a item_value | -r item_value)
```

```
ec2-modify-image-attribute ami_id -p product_code [-p product_code ...]
```

## Description

Modifies an attribute for the specified AMI.

## Attributes

Attribute Name	Type	Description
launchPermission	List	<p>Controls who has permission to launch the AMI. You can grant launch permissions by adding user IDs or make the AMI public by adding the <code>all</code> group. To learn more about sharing AMIs see <a href="#">Sharing AMIs</a>.</p> <div><b>Note</b><p>If another user launches your AMI there is no mechanism to prevent that user from rebundling the image and registering it as a new AMI.</p></div>
productCodes	List	<p>Associates a product code with an AMI. This allows a developer to charge a user for using the AMI.</p> <div><b>Note</b><p>The user must be signed up for the product before they can launch the AMI.</p></div> <p>The product code attribute is a write-once attribute. After a product code is set for an AMI it can not be altered or removed. AMIs are limited to one product code.</p>

## Options

Option	Description	Required
-l	Modifies the <code>launchPermission</code> property. Example: -l	Yes

Option	Description	Required
<code>-a item_value   -r item_value</code>	Adds or removes an attribute item. The type of the item is inferred from the item value. For <code>launchPermission</code> there are two item types: <ul style="list-style-type: none"><li><code>group</code>: The only group currently supported is the <code>all</code> group. Adding this group sets public launch permissions for the AMI.</li><li><code>userId</code>: The user ID is an AWS account ID, without dashes. Adding user IDs grants explicit launch permissions for the AMI.</li></ul> Example: <code>-a all</code>	Yes
<code>-p product_code</code>	Sets the <code>productCodes</code> property. Example: <code>-p 774F4FF8</code>	Yes

## Output

Amazon EC2 returns a table that contains the following information:

- Attribute type identifier.
- ID of the AMI on which attributes are being modified.
- Action performed on the attribute.
- Attribute or attribute list item value type.
- Attribute or attribute list item value.

Amazon EC2 displays errors on `stderr`.

## Examples

The following example modifies launch permission attributes for the `ami-2bb65342` AMI:

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

The following example adds the `774F4FF8` product code to the `ami-2bb65342` AMI:

```
$ ec2-modify-image-attribute ami-2bb65342 -p 774F4FF8
productCodes ami-2bb65342      productCode 774F4FF8
```

## Related Topics

- [ModifyImageAttribute](#)
- [ec2-reset-image-attribute](#)
- [ec2-describe-image-attribute](#)
- [Sharing AMIs](#)

## ec2-reboot-instances

### Syntax

```
ec2-reboot-instances instance_id [instance_id ...]
```

### Description

Reboots one or more specified instances. You must specify at least one instance ID.

### Options

Option	Description	Required
<i>instance_id</i>	Instance identifier that was assigned to the instance at launch. Example: i-3ea74257	Yes

### Output

This command displays no output on success.

Amazon EC2 displays errors on stderr.

### Example

This example reboots the i-3ea74257 instance.

```
$ ec2-reboot-instances i-3ea74257
```

## ec2-register

### Syntax

```
ec2-register manifest
```

### Description

Registers the Amazon Machine Image (AMI) specified in the manifest file and generates a new Amazon Machine Image (AMI) ID.

### Options

Option	Description	Required
<i>manifest</i>	Location and filename of the manifest file. The manifest file must be located in Amazon S3 and must be in the bucket/object form. Example: mybucket/image.manifest.xml	Yes

## Output

Amazon EC2 assigns and returns an AMI ID.

Amazon EC2 displays errors on stderr.

## Example

This example registers the AMI specified in the `image.manifest.xml` manifest file:

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

## Related Topics

- [RegisterImage](#)
- [ec2-deregister](#)
- [ec2-describe-images](#)

# ec2-reset-image-attribute

## Syntax

```
ec2-reset-image-attribute ami_id -l
```

## Description

Resets an attribute for the specified AMI.

The `productCodes` attribute cannot be reset.

## Options

Option	Description	Required
<i>ami_id</i>	The identifier that was assigned to the AMI when it was registered. Example: <code>r-15a4417c</code>	Yes
<code>-l</code>	Resets the <code>launchPermission</code> attribute. Example: <code>-l</code>	Yes

## Output

Amazon EC2 returns a table that contains the following information:

- Attribute type identifier
- ID of the AMI on which the attribute is being reset
- Action identifier ("RESET")

Amazon EC2 displays errors on stderr.

## Example

This example resets the `launchPermission` attribute:

```
$ ec2-reset-image-attribute ami-6ba54002 -l  
launchPermission ami-6ba54002 RESET
```

## Related Topics

- [ResetImageAttribute](#)
- [ec2-modify-image-attribute](#)
- [ec2-describe-image-attribute](#)
- [Sharing AMIs](#)

## ec2-revoke

### Syntax

```
ec2-revoke group [-P 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](#).

### Options

Option	Description	Required
<i>group</i>	The group to which this rule will apply. Example: <code>webserver</code>	Yes
<code>-P protocol</code>	The protocol to revoke. Condition: Applies when specifying a CIDR subnet as the source. Valid Values: <code>tcp</code>   <code>udp</code>   <code>icmp</code> Example: <code>-P tcp</code>	Yes
<code>-p port_range</code>	For the TCP or UDP protocols, this specifies the range of ports to revoke. You can specify a single integer or a range (min-max). Condition: Applies when specifying a CIDR subnet as the source. Example: <code>-p 80</code>	Yes
<code>-t icmp_type_code</code>	For the ICMP protocol, you must specify the ICMP type and code. Use the type:code format, where both are integers. To specify a wildcard for either or both, enter <code>-1</code> .	Yes

Option	Description	Required
	Condition: Applies when specifying a CIDR subnet as the source. Example: <code>-t 2:5</code>	
<code>-u source_group_user</code>	The owner of a group specified using <code>-o</code> . If this is not specified, all groups will refer to the current user. If specified more than once, there must be exactly one <code>-u</code> per <code>-o</code> and each user will be mapped to the corresponding group. Example: <code>-u 495219933132</code>	No
<code>-o source_group</code>	The network source from which traffic will be revoked specified as a security Group. See the description of the <code>-u</code> option for group owner information. Example: <code>-o headoffice</code>	No
<code>-s source_subnet</code>	The network source from which traffic is to be revoked specified as a CIDR subnet range. Example: <code>-s 205.192.8.45/24</code>	No

## Output

A table containing the following information is returned:

- Output type identifier ("GROUP", "PERMISSION").
- Group name. Currently, this will report an empty string.
- Type of rule. Currently, only ALLOW rules are supported.
- Protocol to allow.
- Start of port range.
- End of port range.
- FROM
- Source.

Amazon EC2 displays errors on stderr.

## Example

This example revokes TCP port 80 access from the 205.192.0.0/16 address range for the webserv security group:

```
$ ec2-revoke webserv -P tcp -p 80 -s 205.192.0.0/16
GROUP webserv ""
PERMISSION webserv ALLOWS tcp 80 80 FROM CIDR 205.192.0.0/16
```

## Related Topics

- [RevokeSecurityGroupIngress](#)



- [ec2-add-group](#)
- [ec2-describe-groups](#)
- [ec2-delete-group](#)
- [ec2-authorize](#)

## ec2-run-instances

### Syntax

```
ec2-run-instances ami_id [-n instance_count] [-g group [-g group ...]] [-k keyname] [-d user_data | -f user_data_file] [ --addressing addressing_type] [ --instance-type instance_type]
```

### Description


Launches one or more instances of the specified AMI. Optional parameters include:

- A security group. New instances launch in this group. If no group is specified instances are launched in the default group.
- A key pair name. The public key associated with this key pair name are available to the instances at boot time.
- User data. This data is available to the launched instances. For more information, see [Instance Metadata](#).
- Addressing type. This specifies whether the instance has a NATted address. See [Instance Addressing and Network Security](#) for more information on instance addressing.
- This specifies the instance type.  
Options include `m1.small`, `m1.large`, and `m1.xlarge`.  
The default value is `m1.small`.  
For more information on instance types, see [Selecting Instance Types](#)

If the AMI has a product code attached for which the user has not subscribed, the **ec2-run-instances** call will fail.

### Options

Option	Description	Required
<code>-n <i>instance_count</i></code>	The number of instances to launch. If Amazon EC2 cannot launch the specified number of instances, no instances will launch. If this is specified as a range (min-max), Amazon EC2 will try to launch the maximum number, but no fewer than the minimum number.  Default: 1 Example: <code>-n 5</code>	No
<code>-g <i>group</i></code>	The security group(s) within which to launch the instances. This determines the ingress firewall rules applied to the instances. If you specify more than one group, the security policy will be the union of the security policies of the specified groups.	No

Option	Description	Required
	Default: The default group. Example: <code>-g fooGroup</code>	
<code>-k keyname</code>	The key pair to make available to these instances at boot. Example: <code>-k fooKeyPair</code>	No
<code>-d user_data</code>	Data to make available to the instances. This data is read from the command line of the <code>USER_DATA</code> argument. If you want the data to be read from a file, see the <code>-f</code> option. Example: <code>-d "my user data"</code>	No
<code>-f user_data_file</code>	Data to make available to these instances. The data is read from the file specified by <code>FILE_NAME</code> . To specify user data on the command line, use the <code>-d</code> option. Example: <code>-f data.zip</code>	No
<code>--addressing addressing_type</code>	<p>The address type with which the instance launches. Valid Value: <code>public</code></p> <div>  <p><b>Note</b></p> <p>To support the early beta legacy networking scheme, <code>direct</code> is supported for early beta users. For more information on instance addressing, see <a href="#">Instance Addressing</a>.</p> </div> <p>Example: <code>--addressing public</code></p>	No
<code>--instance-type instance_type</code>	The type of instance to launch. For more information, see <a href="#">Selecting Instance Types</a> . Example: <code>--instance-type m1.small</code>	No

## Output

Amazon EC2 returns a table that contains the following information:

- Output type identifier ("INSTANCE")
- Instance ID which uniquely identifies each running instance.
- AMI ID of the image on which the instance(s) are based.
- DNS name associated with the instance (only present for instances in the `running` state).
- Instance state. This is usually `pending`, which indicates that the instance(s) are preparing to launch.
- Key name. If a key was associated with the instance at launch its name is displayed
- AMI launch index. For more information, see [Instance Metadata](#).
- Instance type. For more information on instance types, see [Selecting Instance Types](#).
- Instance launch time. Specifies when the instance launched.

Amazon EC2 displays errors on stderr.

## Example

This example launches five instances of the ami-6ba54002 AMI:

```
$ ec2-run-instances ami-6ba54002 -n 5
RESERVATION r-0ea54067 495219933132 default
INSTANCE i-3ea74257 ami-6ba54002 pending 0 m1.small 2007-07-11T16:40:44+0000
INSTANCE i-31a74258 ami-6ba54002 pending 1 m1.small 2007-07-11T16:40:44+0000
INSTANCE i-31a74259 ami-6ba54002 pending 2 m1.small 2007-07-11T16:40:44+0000
INSTANCE i-31a7425a ami-6ba54002 pending 3 m1.small 2007-07-11T16:40:44+0000
INSTANCE i-31a7425b ami-6ba54002 pending 4 m1.small 2007-07-11T16:40:44+0000
INSTANCE i-31a7425c ami-6ba54002 pending 5 m1.small 2007-07-11T16:40:44+0000
```

## Related Topics

- [RunInstances](#)
- [ec2-terminate-instances](#)
- [ec2-describe-instances](#)
- [ec2-add-keypair](#)
- [Instance Metadata](#)

## ec2-terminate-instances

### Syntax

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

### Description

Terminates the specified instances.

### Options

Option	Description	Required
<i>instance_id</i>	ID of the instance that was assigned at launch. Example: i-3ea74257	Yes

### Output

Amazon EC2 returns a table that contains the following information:

- Output type identifier ("INSTANCE")
- The instance ID of the instance being terminated.
- The state of the instance prior to being terminated.
- The new state of the instance.

Amazon EC2 displays errors on stderr.

## Example

This example terminates the `i-3ea74257` instance:

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

## Related Topics

- [TerminateInstances](#)
- [ec2-run-instances](#)
- [ec2-describe-instances](#)

# Technical FAQ

## Topics

- [General Information](#)
- [Operation Information](#)
- [Instance Types and Architectures](#)
- [IP Information](#)
- [Monitoring, Errors, and Unexpected Behavior](#)
- [Error Messages](#)
- [Paid AMIs](#)
- [Miscellaneous](#)

This section contains answers to commonly asked questions.

## General Information

*How many instances can I launch?*

Each user has a concurrent running instance limit. For new users, this limit is 20. If you need more than 20 instances, please contact [aws@amazon.com](mailto:aws@amazon.com).

*How do I sign a request?*

Information on signing SOAP requests is provided in [Request Authentication](#). Information on signing Query requests is provided in [Query API Authentication](#)

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

When you sign up with Amazon Web Services, you are provided an AWS Account ID. This is your username. For more information, refer to the *Amazon Elastic Compute Cloud Getting Started Guide*.

*Why do my instances take so long to start?*

Amazon EC2 must move the images around the network before they can be launched. For big images and/or congested networks, this can take several minutes. To improve performance, images are cached.

As you launch your images more frequently, it should be less noticeable.

*How durable are the instance stores?*

Instance stores appear to an instance as a local disk. They will survive intentional and unintentional reboots of the instance unless the instance terminates or the underlying drive fails.

You should always backup or replicate important data.

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

The instances will terminate and will need to be relaunched. The data on the instances' hard drives will be lost.

Always replicate important data or store it in Amazon S3.

*Can I use my own kernel?*

Not at present.

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

Yes. For more information, see [Selecting Instance Types](#).

*Is there a REST interface to Amazon EC2?*

Not at present. You can use the SOAP API, Query API, or the command-line tools.

*How does Amazon EC2 handle load balancing?*

With a service as flexible as Amazon EC2, you can use many types of load balancing systems. The load balancing instances can forward traffic to other systems. There are several open source solutions that are in wide use.

*Does Amazon perform system maintenance?*

Yes. Periodically, Amazon might perform maintenance that requires a reboot of your system. Make sure your instances can recover and restart after being rebooted.

## Operation Information

*How do I handle time synchronization between instances?*

You can set up NTP (Network Time Protocol). For more information, go to <http://-www.ntp.org/>. NTP is particularly important if you plan on using any Amazon web services (such as Amazon S3 or Amazon EC2) from within an instance, since requests to these services must be timestamped.

*Is there a method for an instance to discover its own instance ID?*

From within your instance you can use REST-like queries to <http://169.254.169.254/2007-08-29/> to retrieve various instance-specific metadata, including the instance ID. For more information, see [Instance Metadata](#).

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

Yes, although the size of the data is limited to 16K. For more information, see [Instance Metadata](#).

*Is there a way to run a script on instance termination?*

Not with any reliability. Amazon EC2 tries to shut an instance down cleanly (running system shutdown scripts), but there is only a short time available. In some cases (e.g., hardware failure), this does not happen.

Since there is no way to ensure shutdown scripts run, have a strategy to deal with abnormal terminations.

*How can I allow other people to launch my AMIs?*

You can allow other users to launch your AMIs by modifying the AMI's `launchPermission` attribute. You can grant public launch permissions or explicit permissions to specific users.

To grant public launch permissions:

```
PROMPT> ec2matt ami-2bb65342 -t launchPermission -a -i group=all
```

To grant a specific user launch permissions:

```
PROMPT> ec2matt ami-2bb65342 -t launchPermission -a -i userId=495219933132
```

To clear additional launch permissions for an AMI:

```
PROMPT> ec2ratt ami-2bb65342 -t launchPermission
```

*Why do I need to reregister a rebundled AMI? Can I keep the same AMI ID?*

An AMI ID is associated with the physical bits in an image. To protect users from images being modified, we require you to reregister AMIs after rebundling.

*Can I pass JVM properties to the command line tools?*

Yes. By setting the environment variable `EC2_JVM_ARGS`, you can pass arbitrary JVM properties to the command line tools.

*Can I use a proxy with the command line tools?*

Yes. By passing in JVM properties through the `EC2_JVM_ARGS` environment variable, you can specify proxy settings for the command line tools. For example, in Linux:

```
export EC2_JVM_ARGS="-Dhttp.proxyHost=http://my.proxy.com -Dhttp.proxyPort=8080"
```

The following properties are supported for configuring a proxy:

Setting	Description
https.proxyHost	HTTPS proxy host
https.proxyPort	HTTPS proxy port
http.proxyHost	HTTPS proxy host
http.proxyPort	HTTPS proxy port
http.proxyRealm	Proxy realm (https and http)
http.proxyUser	Proxy username (https and http)
http.proxyPass	Proxy password (https and http)



#### Note

`https.proxyHost` should be used when `EC2_URL` points to an https host, and `http.proxyHost` when `EC2_URL` points to an http host.

## Instance Types and Architectures

*What happened to the original instance type?*

The original instance type is still available. It is called the small instance (m1.small) and it has the same technical specifications.

*Will the original instance type be retired soon?*

There are no plans to retire the original instance type.

*If I do not specify an instance type at launch, what type of instance will I get?*

You will get a small instance instance type which is equivalent to the original Amazon EC2 instance type.

*Does my instance limit apply to all instance types or is there a separate limit for each type?*

The instance limit applies to the sum of all instances, regardless of type. There is no separate instance limit per type.

*Can I mix instance types, or do I have to use the same type for all of my instances?*

You can launch any combination of instance types. Choose the instance types that have the most appropriate memory, CPU, and storage for each function within your application.

*Which instance types are 32-bit and which are 64-bit?*

The small Instance (m1.small) is 32-bit. The large (m1.large) and extra large (m1.xlarge) instances are 64-bit.

*Can I launch any AMI on any type of instance?*

No. You must use 64-bit AMIs on large (m1.large) and extra large (m1.xlarge) instances and 32-bit AMIs on small Instances (m1.small).

*Do I have to do anything special to bundle the large or extra large instances?*

Make sure to use the latest AMI Tools.

*Can I build an AMI that works on both 32-bit and 64-bit instances?*

No, an AMI is either a 32-bit AMI or a 64-bit.

*Can I run 32-bit applications on 64-bit AMIs?*

You can run a 32-bit application on a 64-bit host if the Linux kernel is compiled with IA32 emulation and the correct 32-bit libraries are available.

By default, the Amazon DomU Kernel has IA32 emulation enabled and there are many public AMIs that include pre-installed 32-bit libraries. If the library you require is not included with the AMI, you can install it using standard tools (e.g., yum).



*How fast is the disk?*

The large and extra large instances have higher and more consistent I/O performance than the original (small) instance.



**Note**

The first write to any given block of the disk will be slower than subsequent writes. For more information, see [Disk Performance Optimization](#)

*Can I RAID the spindles exposed on large and extra large instances?*

Yes, you can use software RAID on top of the exposed spindles.



**Note**

The initial RAID setup might take a long time. For more information, see [Disk Performance Optimization](#)

*Can I launch the large or extra large instance in direct addressed space?*

No, the large and extra large instances are only available as NAT addressed instances. For more information, see [Instance Addressing and Network Security](#).

## IP Information

*How do I host a public domain if I have to DHCP an IP address?*

You can use a dynamic DNS service, such as [DynDNS](#) or [ZoneEdit](#).

*Why can't I connect to my instances public IP address from another instance?*

There is a known limitation that prevents direct-addressed instances from contacting NAT-addressed instances through their public IP address. You should use the internal IP/DNS name when communicating between Amazon EC2 instances. This will ensure the lowest latency, highest throughput, and lowest cost network route.

*Why do I get an internal (RFC 1918) IP address when I look up a DNS name that I expect to map to my instance's external IP address?*

Amazon EC2 instances cannot currently access other instances in the new NAT environment using their public NAT IP address. Instead, they must use the private address. Traffic originating from the Internet must use the public NAT IP address. Within Amazon EC2, DNS requests for the external DNS name of an instance will resolve to the internal IP address of the corresponding instance

*Can I use IPv6?*

Amazon EC2 presently supports the ICMP, TCP and UDP protocol families of IPv4-based networking.

*Why is Amazon EC2 Using NAT?*

Public IP space is a limited resource. Amazon EC2 is adopting NAT to assure that we are able to efficiently make use of our public internet addresses.

Furthermore, the new NAT networking will enable Amazon to deliver new features in the future. For

example, some users do not want external addresses. This would allow for non-internet routable clusters, which will further preserve IPs and increase security for those not running public facing servers.

*Can I use a static IP in my instances?*

Not at present. Your image must be configured as a DHCP client and it will be assigned an IP address. Currently, all instances come with internet- addressable IP addresses. If you enable access through the firewall from the "world", you can address them from anywhere.

*How does the instance know its public and private addresses?*

From within the instance, issue the following HTTP queries:

To obtain the internal IP address:

```
curl http://169.254.169.254/2007-08-29//meta-data/local-ipv4
```

To obtain the public IP address:

```
curl http://169.254.169.254/2007-08-29//meta-data/public-ipv4
```

## Monitoring, Errors, and Unexpected Behavior

*How do I monitor my systems?*

Amazon EC2 provides basic monitoring. You can use `DescribeInstances` to check whether an instance appears to be running. However, if you are using Amazon EC2 as your data center, you might want to set up for sophisticated monitoring on your instances, such as SNMP.

*Why can't I "talk" to my instances?*

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

Amazon EC2 changes the state of your instance to `running` after your operating system starts booting. Depending on your AMI, there will be a delay before the instance is fully set up and functional.

If your instance has been running for several minutes, you verify you authorized the appropriate access to your host through the Amazon EC2 firewall. If you have launched your instances without specifying a security group, the `default` group is used. Permissions on the `default` group are very strict and disallow all access from the Internet and other groups. You will need to modify the permissions of your `default` group or set up a new group with appropriate permissions. For more information, see [Network Security](#)

If this doesn't solve your issue, make sure you authorized port 22 and try to open an SSH connection with verbose output. Use the man page for the exact syntax of your system, but the command is likely to be similar to `ssh -vv root@[hostname]`. This output is very useful if you are posting to the forum.

*Why did my instance terminate immediately after launch?*

Launch errors can be the result of an internal error during launch or a corrupt Amazon EC2 image. Internal errors are rare, as we actively test for and isolate suspect hosts. Consult the `DescribeInstances` operation for details on why your instance failed to launch.



### Note

The `ec2-describe-instances` command line tool does not provide this information.

Use the `-v` flag to read the detailed SOAP response and get detailed information.

You can also attempt to launch the image again. If this proves to be a persistent problem (especially with a shared image), post to the [AWS forums](#).

I ran `shutdown` from within an `ssh` session, but my instance still shows up as running when I query it with `DescribeInstances` and I can't shell into it.

This is a "feature" of the `shutdown` command. If you issue `shutdown` without a `-h` (halt) flag, it shuts down the network and switches to single user mode. The instance is still running, but without a network.

You should always use `shutdown -h` when working inside an Amazon EC2 instance.

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

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

This situation is rare and might be the result of a software error or misconfiguration.

We actively monitor for this; please contact us if it occurs.

*Why do I get an "AuthFailure: User is not AMI creator" error when I try to register an image?*

Make sure that you are using the correct user ID and certificate to create and upload the image. You must use the same ID and certificate to register the image with Amazon EC2.

## Error Messages

*Why do I get an "InsufficientInstanceCapacity" error when I try to launch an instance?*

This error indicates that we do not currently have enough available capacity to service your request.

If you are requesting a large number of instances, there might not be enough server capacity to host them. You can try again later or specify a smaller number of instances.

*Why do I get an "InstanceLimitExceeded" error when I try to launch an instance?*

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

If you need additional capacity, please contact us at [aws@amazon.com](mailto:aws@amazon.com).

*Why can't I retrieve my instance-specific data from within a running instance when querying `http://169.254.169.254/2007-08-29/`?*

The Parameterized Launches feature is available to instances that were launched after the feature was released. If you launched your instance before this, the data will not be available. If you want to use this functionality, relaunch your instances.

If, after relaunching your instance, you still experience problems retrieving the data, check the following:

- Are you using the correct base URI (`http://169.254.169.254/2007-08-29/`)
- Are you using the correct URI for the data you're trying to retrieve? Depending on the data you're trying to retrieve, a trailing `'/'` might be required.

- Did you specify any launch data when launching your instances? If not you will get a HTTP error response (404) when trying to retrieve the user data.



#### Note

Instance metadata is always available, even if you do not specify it at instance launch.

*Why do I get keep getting "Request has expired" errors?*

To reduce the risk of replay attacks, our requests include a timestamp. This and the most important parts of the request are signed to ensure the message (including the timestamp) cannot be modified without detection.

If the difference between the timestamp in the request and the time on our servers is larger than 5 minutes, the request is too old (or too new) and an error is returned.

You need to ensure that your system clock is accurate and configured to use the correct timezone. For more information, go to [NTP](#).

## Paid AMIs



#### Note

You can still share AMIs without charging. Public and paid AMIs can be listed in the Resource Center.

*How can I determine if a particular AMI is a paid AMI?*

By describing images (ec2dim) with the "-a" flag and looking for AMIs that have a product code. For example, if you run `ec2dim -a`, the result contains an AMI with the ID `ami-bd9d78d4`. This is our Demo Paid AMI with product code `A79EC0DB`.

*How can I determine if a public AMI is paid?*

By describing images (ec2dim). An AMI is a paid AMI if a product code is returned. Example: run `ec2dim -a amazon`, and the AMI `ami-bd9d78d4` will be returned with a product code (`A79EC0DB`).

*Is there anything that prevents a paid AMI from being rebundled? How can this be restricted?*

Paid AMIs are comparable to shared AMIs with regards to rebundling and trying to restrict rebundling. If you allow a user running the AMI to see all of its contents (e.g. by giving root access to the AMI), the user could rebundle these into their own AMI.

*Why can't I query a particular AMI's attributes to see if the AMI is paid?*

Only the owner of an AMI can query the AMI attributes. However, anyone can tell if an AMI is paid by describing images (ec2dim). An AMI is paid if a product code is returned. Example: run `ec2dim -a amazon`, and the AMI with ID `ami-bd9d78d4` will be returned with a product code (`A79EC0DB`).

*Who can use the confirm-product-instance command?*

Only the owner of the AMI can use this command. Owners use this command with supported AMIs to determine if a supported instance with a given product code attached is up and running.

*Will the product code be inherited by the rebundled AMI?*

If your customer uses AWS tools to rebundle the AMI, the product code associated with the AMI is inherited by the rebundled AMI. When launching the rebundled AMI the customer is still billed for usage based on your price.



**Note**

This is a convenience feature and not a guarantee that the product code will always be attached to rebundled AMIs.

Note that the customer's workflow could bundle the AMI outside of Amazon EC2, or the customer could use modified versions of the AWS tools, preventing the product code from being inherited.

*Will the kernel/ramdisk be inherited by the rebundled AMI?*

If the AMI is rebundled on an instance that is running with a certain kernel/ramdisk, then the kernel/ramdisk will be inherited by the rebundled AMI.



**Note**

This is a convenience feature and not a guarantee that the kernel/ramdisk will always be attached to rebundled AMIs.

*Who can use the `confirm-product-instance` command?*

If the AMI is rebundled on an instance that is running with a certain kernel/ramdisk, then the kernel/ramdisk will be inherited by the rebundled AMI.



**Note**

This is a convenience feature and not a guarantee that the product code will always be attached to rebundled AMIs.

## Miscellaneous

*Are there any special requirements to use FTP?*

The File Transfer Protocol (FTP) has a PORT command by which a client sends its address back to the server. The server then connects to the client at that address to send the file data. If the client looks up its own internal address and sends this to the server, the connection will fail. In this specific case, there are two solutions to the problem. First, configure the client to send its public IP address. Second, the client can use "passive FTP" which makes connections only to the server, rather than from the server to the client. In general, applications which encode local addresses and port numbers in data sent to external servers might have problems with NAT. Care must always be taken to send the public address, rather than the internal one.

We recommend using passive mode unless it is not supported by the FTP server.

# Glossary

Amazon machine image (AMI)	An Amazon Machine Image (AMI) is an encrypted machine image stored in Amazon S3. It contains all the information necessary to boot instances of your software.
explicit launch permission	Launch permission granted to a specific user.
instance	Once an AMI has been launched, the resulting running system is referred to as an instance. All instances based on the same AMI start out identical and any information on them is lost when the instances are terminated or fail.
group	Also known as a security group, groups define firewall rules that can be shared among a group of instances that have similar security requirements. The group is specified at instance launch.
launch permission	AMI attribute allowing users to launch an AMI
public AMI	An AMI that all users have launch permissions for.
reservation	A collection of instances started as part of the same launch request.
shared AMI	AMIs that developers build and make available for other AWS developers to use.

# Document Conventions

This section establishes the common typographical and symbol use conventions for AWS technical publications.

## Typographical Conventions

This section describes common typographical use conventions.

Convention	Description/Example
Call-outs	<p>A call-out is a number in the body text to give you a visual reference. The reference point is for further discussion elsewhere.</p> <p>You can use this resource regularly. <b>1</b></p>
Code in text	<p>Inline code samples (including XML) and commands are identified with a special font.</p> <p>You can use the command <code>java -version</code>.</p>
Code blocks	<p>Blocks of sample code are set apart from the body and marked accordingly.</p> <pre># ls -l /var/www/html/index.html -rw-rw-r-- 1 root root 1872 Jun 21 09:33 / var/www/html/index.html # date Wed Jun 21 09:33:42 EDT 2006</pre>
Emphasis	<p>Unusual or important words and phrases are marked with a special font.</p> <p>You <i>must</i> sign up for an account before you can use the service.</p>
Internal cross references	<p>References to a section in the same document are marked.</p>

Convention	Description/Example
	See <a href="#">Document Conventions</a> .
Logical values, constants, and regular expressions, abstracta	<p>A special font is used for expressions that are important to identify, but are not code.</p> <p>If the value is <code>null</code>, the returned response will be <code>false</code>.</p>
Product and feature names	<p>Named AWS products and features are identified on first use.</p> <p>Create an <i>Amazon Machine Image</i> (AMI).</p>
Operations	<p>In-text references to operations.</p> <p>Use the <code>GetHITResponse</code> operation.</p>
Parameters	<p>In-text references to parameters.</p> <p>The operation accepts the parameter <i>AccountID</i>.</p>
Response elements	<p>In-text references to responses.</p> <p>A container for one <code>CollectionParent</code> and one or more <code>CollectionItems</code>.</p>
Technical publication references	<p>References to other AWS publications. If the reference is hyperlinked, it is also underscored.</p> <p>For detailed conceptual information, see the <i>Amazon Mechanical Turk Developer Guide</i>.</p>
User entered values	<p>A special font marks text that the user types.</p> <p>At the password prompt, type <b>MyPassword</b>.</p>
User interface controls and labels	<p>Denotes named items on the UI for easy identification.</p> <p>On the <b>File</b> menu, click <b>Properties</b>.</p>
Variables	<p>When you see this style, you must change the value of the content when you copy the text of a sample to a command line.</p> <p><code>% ec2-register &lt;your-s3-bucket&gt;/image.manifest</code></p> <p>See also the symbol convention below.</p>

## Symbol Conventions



This section describes the common use of symbols.

Convention	Symbol	Description/Example
Mutually exclusive parameters	(Parentheses   and   vertical   bars)	Within a code description, bar separators denote options from which one must be chosen.  <pre>% data = hdfread (start   stride   edge)</pre>
Optional parameters XML variable text	[square brackets]	Within a code description, square brackets denote completely optional commands or parameters.  <pre>% sed [-n, -quiet]</pre> Use square brackets in XML examples to differentiate them from tags.  <pre>&lt;CustomerId&gt;[ ID]&lt;/Custom erId&gt;</pre>
Variables	<arrow brackets>	Within a code sample, arrow brackets denote a variable that must be replaced with a valid value.  <pre>% ec2-register &lt;your-s3-bucket&gt;/image.m anifest</pre>