Amazon Elastic Compute Cloud

User Guide

API Version 2012-07-20

Amazon Web Services

Amazon Elastic Compute Cloud: User Guide

Amazon Web Services

Amazon Elastic Compute Cloud User Guide

This is the User Guide for Amazon Elastic Compute Cloud (Amazon EC2). Amazon EC2 is a web service that provides resizeable computing capacity—literally servers in Amazon's data centers—that you use to build and host your software systems.

This guide picks up where the Amazon Elastic Compute Cloud Getting Started Guide leaves off. It helps you understand the components and features that Amazon EC2 provides and how to use them. You'll learn how to access Amazon EC2 through a web-based GUI, command line tools, and the Amazon EC2 API.

How Do I...?

How Do I?	Relevant Topics
Get a general overview of the product and information about pricing	Amazon EC2 product page
Get a quick, hands-on introduction to Amazon EC2	Amazon Elastic Compute Cloud Getting Started Guide
Get a quick summary of the basic infrastructure components that Amazon EC2 provides	What is Amazon EC2? (p. 3)
Get detailed information about how to use the Amazon EC2 components and features, with instructions for several different interfaces	Amazon Machine Images (AMI) (p. 23) Amazon EC2 Instances (p. 84) Amazon Elastic Block Store (Amazon EBS) (p. 434) Network and Security (p. 349)
Get information specific to using Amazon EC2 with the Windows Server operating system.	Amazon Elastic Compute Cloud Microsoft Windows Guide

How Do I?	Relevant Topics
Start using the command line tools (also known as the Amazon EC2 API tools)	Setting Up the Amazon EC2 Command Line Tools (p. 522) Amazon Elastic Compute Cloud Command Line Reference
Start using the Query or SOAP API	Making API Requests (p. 532) Amazon Elastic Compute Cloud API Reference
Find libraries to access Amazon EC2	Available Libraries (p. 539)

Note

The content from the original *Amazon Elastic Compute Cloud Developer Guide* is included in this guide.

The Filter View Menu

The procedures in the Amazon EC2 User Guide include instructions for the AWS Management Console, the command line interface (CLI) tools, and the EC2 Query API. In the HTML version of this document, you can show instructions for all available interfaces or a single interface. There is a **Filter View** menu in the upper-right corner of pages that include instructions for multiple interfaces. Select the interface of your choice to hide the other interfaces, or select **All** to show the instructions in all available interfaces.

Filter View: All ▾
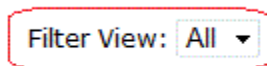
Additional Resources

- Getting Started with AWS

What is Amazon EC2?

Topics

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizeable computing capacity—literally, servers in Amazon's data centers—that you use to build and host your software systems. You can access the components and features that EC2 provides using a web-based GUI, command line tools, and APIs.

With EC2, you use and pay for only the capacity that you need. This eliminates the need to make large and expensive hardware purchases, reduces the need to forecast traffic, and enables you to automatically scale your IT resources to deal with changes in requirements or spikes in popularity related to your application or service.

Main Components of EC2

You might be considering creating a new application to run in the cloud, or moving an existing application from your own servers into the cloud. To do either, you should understand the infrastructure available in the cloud and how it's similar or different from your own data centers. This section gives a brief description of the main components that EC2 provides.

Topics

Amazon Machine Images and Instances

An *Amazon Machine Image (AMI)* is a template that contains a software configuration (operating system, application server, and applications). From an AMI, you launch *instances*, which are running copies of the AMI. You can launch multiple instances of an AMI, as shown in the following figure.
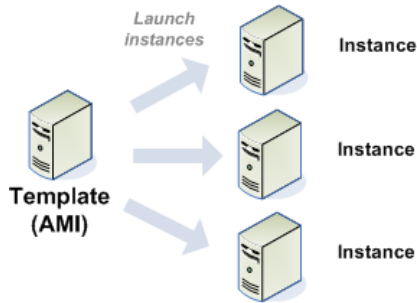
Your instances keep running until you stop or terminate them, or until they fail. If an instance fails, you can launch a new one from the AMI.

You can use a single AMI or multiple AMIs depending on your needs. From a single AMI, you can launch different *types of instances*. An instance type is essentially a hardware archetype. As illustrated in the following figure, you select a particular instance type based on the amount of memory and computing power you need for the application or software that you plan to run on the instance. For more information about the available instance types, see Instance Families and Types (p. 85).
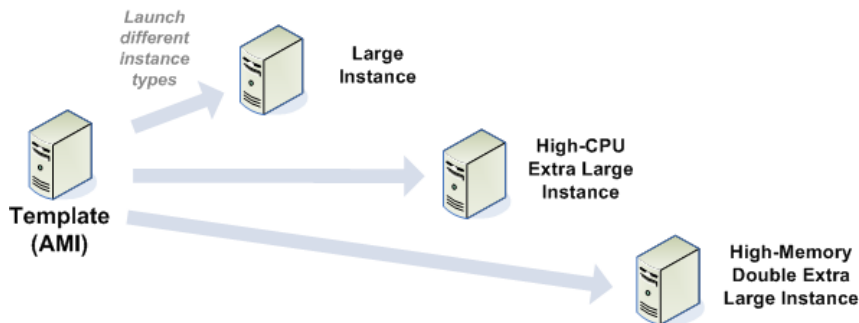
Amazon publishes many AMIs that contain common software configurations for public use. In addition, members of the AWS developer community have published their own custom AMIs. For more information, go to Amazon Machine Images (AMIs).

You might only need to use AMIs that Amazon or other reputable sources provide, and you can simply customize the resulting instances (e.g., run a script) to provide the data or software you need each time you launch an instance. You can also create your own custom AMI or AMIs; then you can run your application by launching one of your custom AMIs.

For example, if your application is a web site or web service, your AMI could be preconfigured with a web server, the associated static content, and the code for all dynamic pages. Alternatively, you could configure your AMI to install all required software components and content itself by running a bootstrap script as soon as the instance starts. As a result, after launching the AMI, your web server will start and your application can begin accepting requests.

For information about AMIs and instances, see Amazon Machine Images (AMI) (p. 23) and Amazon EC2 Instances (p. 84).

Regions and Availability Zones

Amazon has data centers in different areas of the world (for example, North America, Europe, nd Asia). Correspondingly, Amazon EC2 is available to use in different *Regions*. By launching instances in separate Regions, you can design your application to be closer to specific customers or to meet legal or other requirements. Prices for Amazon EC2 usage vary by Region (for more information about pricing by Region, go to the Amazon EC2 Pricing page).

Each Region contains multiple distinct locations called *Availability Zones* (illustrated in the following diagram). Each Availability Zone is engineered to be isolated from failures in other Availability zones and to provide inexpensive, low-latency network connectivity to other zones in the same Region. By launching instances in separate Availability Zones, you can protect your applications from the failure of a single location.
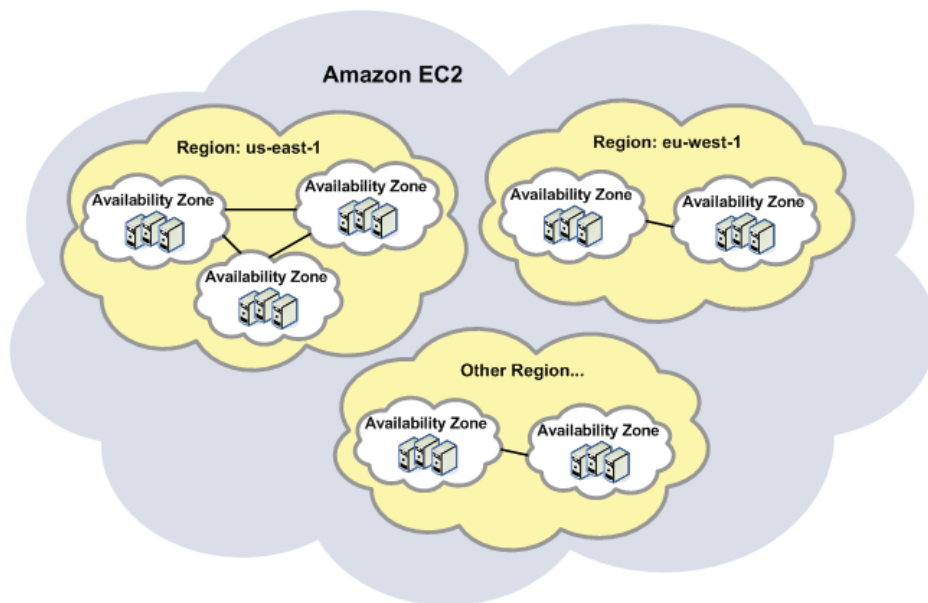
For more information about the available Regions and Availability Zones, see Regions and Availability Zones (p. 111).

Storage

When using an instance, you might have data that you need to store. Amazon EC2 offers the following storage options:

- Amazon Elastic Block Store (Amazon EBS)
- Amazon EC2 Instance Store (p. 479)
- Amazon Simple Storage Service (Amazon S3)

Amazon EBS

Amazon EBS volumes are the recommended storage option for the majority of use cases. Amazon EBS provides your instances with persistent, block-level storage. Amazon EBS volumes are essentially hard disks that you can attach to a running instance.

Amazon EBS is particularly suited for applications that require a database, file system, or access to raw block-level storage.
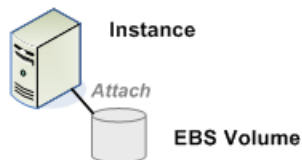
You can attach multiple volumes to an instance. To keep a back-up copy, you can create a snapshot of the volume. As illustrated in the following figure, snapshots are stored in Amazon S3.
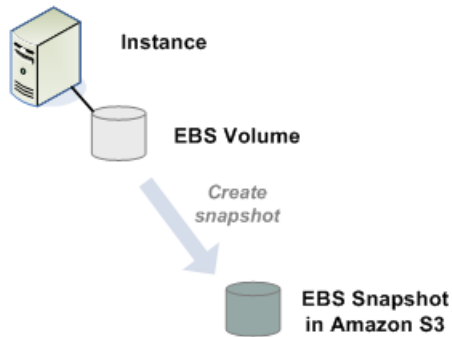
You can create a new Amazon EBS volume from a snapshot, and attach it to another instance, as illustrated in the following figure.
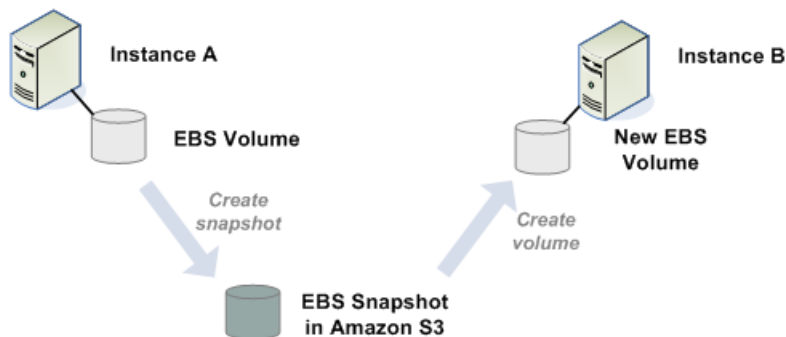
You can also detach a volume from an instance and attach it to a different one, as illustrated in the following figure.

For more information about Amazon EBS volumes, see Amazon Elastic Block Store (p. 434).

Instance Store

All instance types, with the exception of Micro instances, offer instance store. This is storage that doesn't persist if the instance is stopped or terminated. For more information, see Amazon EC2 Instance Storage (p. 479).

Instance store is an option for inexpensive temporary storage. You can use instance store volumes if you don't require data persistence.

Amazon S3

Amazon S3 is storage for the Internet. It provides a simple web service interface that enables you to store and retrieve any amount of data from anywhere on the web. For more information about Amazon S3, see the Amazon S3 product page.

Root Device Storage

When you launch an Amazon EC2 instance, the root device contains the image used to boot the instance. For more information, see Root Device Volume (p. 117).

When Amazon EC2 was first introduced, all AMIs were backed by instance store, which means that the root device for an instance launched from the AMI is an instance store volume. When we introduced Amazon EBS, we also introduced AMIs that are backed by Amazon EBS, which means that the root device for an instance launched from the AMI is an Amazon EBS volume. The description of an AMI includes the type of root device (you'll see either *ebs* or *instance store*). This is important because there are significant differences in what you can do with each type of AMI. For a discussion of these differences, see Storage for the Root Device (p. 24).

Databases

The application you're running on EC2 might need a database. The following are common ways to implement a database for your application:

- Use Amazon Relational Database Service (Amazon RDS), which enables you to easily get a managed relational database in the cloud
- Launch an instance of a database AMI, and use that EC2 instance as the database

Amazon RDS offers the advantage of handling your database management tasks, such as patching the software, backing up and storing the backups, etc. For more information about Amazon RDS, go to the Amazon RDS product page.

Networking and Security

Each instance is launched into the Amazon EC2 network space and assigned a public IP address. Instances can fail or terminate for reasons outside of your control. If one fails and you launch a replacement

instance, the replacement will have a different public IP address than the original. However, your application might need a static IP address. Amazon EC2 offers *elastic IP addresses* for those situations. For more information, see Instance IP Addresses (p. 383).

You use *security groups* to control who can access your instances. These are analogous to an inbound network firewall that allows you to specify the protocols, ports, and source IP ranges that are allowed to reach your instances. You can create multiple security groups and assign different rules to each group. You can then assign each instance to one or more security groups, and we use the rules to determine which traffic is allowed in to the instance. You can configure a security group so that only specific IP addresses or specific security groups have access to the instance.

The following figure shows a basic three-tier web-hosting architecture running on Amazon EC2 instances. Each layer has a different security group (indicated by the dotted line around each set of instances). The security group for the web servers only allows access from hosts over TCP on ports 80 and 443 (HTTP and HTTPS) and from instances in the *App Servers* security group on port 22 (SSH) for direct host management.

The security group for the app servers allows access from the *Web Servers* security group for web requests, and from your corporate subnet over TCP on port 22 (SSH) for direct host management. Your support engineers could log directly into the application servers from the corporate network, and then access the other instances from the application server boxes.

The *DB Servers* security group permits only the App Servers security group to access the database servers.
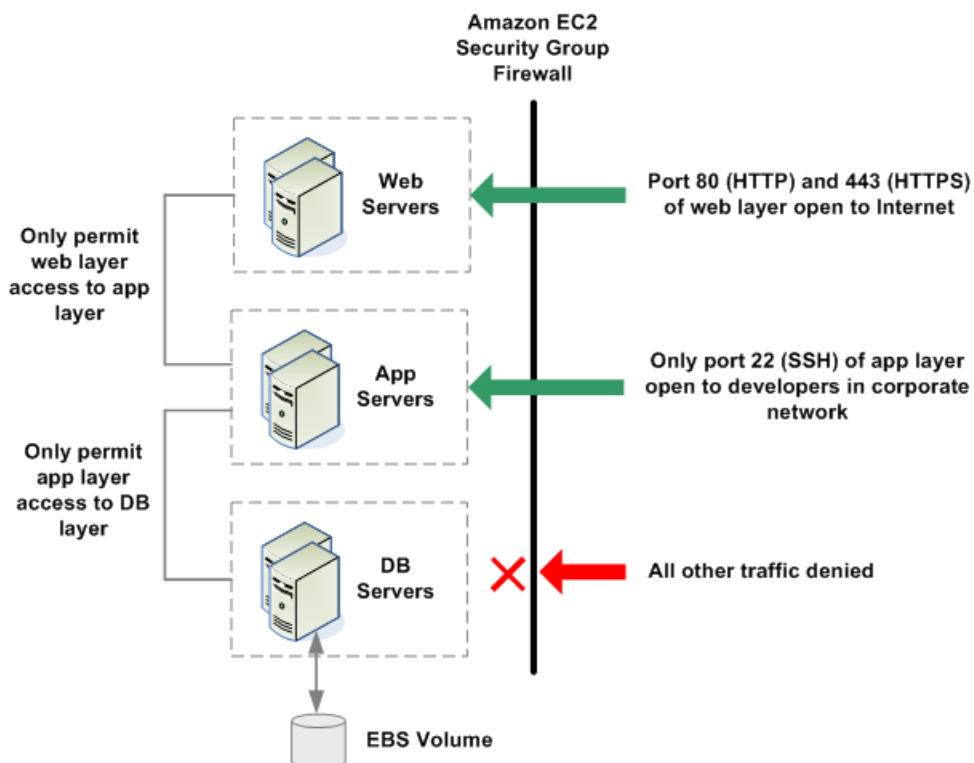
For more information about security groups, see Security Groups (p. 416).

Monitoring, Auto Scaling, and Load Balancing

AWS provides several features that enable you to do the following:

- Monitor basic statistics for your instances and Amazon EBS volumes
  For more information, see Monitoring Your Instances and Volumes with CloudWatch (p. 280).

- Automatically scale your EC2 capacity up or down according to conditions you define
  For more information, go to the Auto Scaling Developer Guide.

- Automatically distribute incoming application traffic across multiple EC2 instances
  For more information, go to the Elastic Load Balancing Developer Guide.

AWS Identity and Access Management

Amazon EC2 integrates with AWS Identity and Access Management (IAM), a service that lets your organization do the following:

- Create users and groups under your organization's AWS account
- Easily share your AWS account resources between the users in the account
- Assign unique security credentials to each user
- Granularly control users access to services and resources
- Get a single AWS bill for all users under the AWS account

For example, you can use IAM with Amazon EC2 to control which users under your AWS account can create AMIs or launch instances.

For general information about IAM, go to:

- Identity and Access Management (IAM)
- AWS Identity and Access Management Getting Started Guide
- Using AWS Identity and Access Management

For specific information about how you can control User access to Amazon EC2, go to Integrating with Other AWS Products in *Using AWS Identity and Access Management*.

Available EC2 Interfaces

Topics
- AWS Management Console (p. 9)
- Command Line Tools (API Tools) (p. 10)
- Programmatic Interfaces (p. 10)

AWS provides different interfaces to access EC2.

AWS Management Console

The AWS Management Console is a simple web-based GUI (see the following screenshot). For more information about using the console, go to the Amazon Elastic Compute Cloud Getting Started Guide.
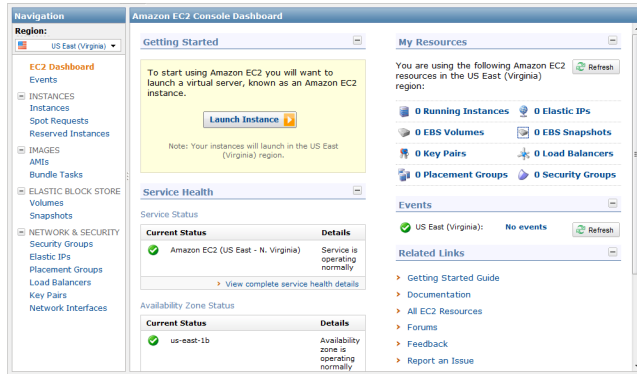
Command Line Tools (API Tools)

EC2 provides a Java-based command-line client that wraps the EC2 SOAP API. For more information, see Setting Up the Amazon EC2 Command Line Tools (p. 522) and Amazon Elastic Compute Cloud Command Line Reference.

Programmatic Interfaces

The following table lists the programmatic interfaces you can use to access EC2.

Type of Access	Description
AWS SDKs	AWS provides the following SDKs: • AWS SDK for Java • AWS SDK for .NET • AWS SDK for PHP
Third-Party Libraries	Developers in the AWS developer community also provide their own libraries, which you can find at the following AWS developer centers: • AWS Java Developer Center • AWS PHP Developer Center • AWS Python Developer Center • AWS Ruby Developer Center • AWS Windows and .NET Developer Center
EC2 API	If you prefer, you can code directly to the EC2 API (Query or SOAP). For more information, see Making API Requests (p. 532), and go to Amazon Elastic Compute Cloud API Reference.

How You're Charged for EC2

With EC2, you pay for only what you use, and there's no minimum charge. Your charges are broken down into these general parts:

- Instance usage
- Data transfer
- Storage

For a complete list of charges and specific prices, go to the Amazon EC2 pricing page.

To see your bill, go to your AWS Account Activity page. Your bill contains links to usage reports that provide details about your bill.

Canceling Amazon EC2

If you decide that Amazon EC2 isn't what you need, you can cancel your registration at any time.

To cancel Amazon EC2

1. Go to http://aws.amazon.com.
2. Click **My Account/Console**, and then click **Account Activity**.
3. Sign in to Amazon Web Services, if you haven't already.
4. Under **Account**, click **Manage Your Account**.
5. Under **Cancel Selected Services** page, select **Elastic Compute Cloud**, and then click **Cancel Service**.

What's Next?

This section introduced you to the basic infrastructure components that EC2 offers. What should you do next?

Get a Hands-On Introduction to EC2

If you haven't walked through the Amazon Elastic Compute Cloud Getting Started Guide, we recommend you do that next. In that guide, you'll do a quick hands-on exercise where you launch an instance and connect to it.

Understand Differences Between the Cloud and Your Data Center

You need to understand the key differences between running your application on infrastructure in the cloud versus on servers in your own data center. For more information, go to the technical whitepaper: Architecting for the Cloud: Best Practices.

Start Thinking about Instance Failure and Fault Tolerance

It's important for you to design your application to handle the failure of a cloud infrastructure component. For example, it's inevitable that EC2 instances will eventually fail, so you need to plan for it. An instance failure isn't a problem if your application is designed to handle it. For more information, see the technical whitepaper: Building Fault-Tolerant Applications on AWS.

For a complete list of the AWS whitepapers, go to AWS Cloud Computing Whitepapers page.

Within this guide, you should read the following sections to understand features of EC2 that help you build fault tolerant applications:

- Using Elastic IP Addresses (p. 401)
- Amazon Elastic Block Store (p. 434)
- Auto-Scaling and Load Balancing Your Instances (p. 277)
- Monitoring Your Instances and Volumes with CloudWatch (p. 280)

Learn More about EC2

The main sections in this guide describe in more detail the technical aspects of the infrastructure components that were briefly described in the preceding sections. We recommend you understand how these components work before designing your application or service to run on EC2.

Getting Started with Amazon EC2 Linux Instances

To get started using Amazon Elastic Compute Cloud (Amazon EC2) Linux instances, complete the steps shown in the following table. You'll primarily use the AWS Management Console, a point-and-click web-based interface.

Tip

This tutorial has you launch and connect to an Amazon EC2 Linux instance. If you'd prefer to launch and connect to an Amazon EC2 Windows instance, see this tutorial: Getting Started with Amazon EC2 Windows Instances.

For more information about Amazon EC2, see the Amazon EC2 product page and What is Amazon EC2? (p. 3) in the Amazon EC2 User Guide.

To get started with Amazon EC2

Step	Description
Step 1: Sign Up for Amazon EC2 (p. 15)	Create an AWS account, if you don't have one already.

Step	Description
Step 2: Launch an Amazon EC2 Instance (p. 15)	Launch an Amazon Linux instance which is derived from this AMI.
Step 3: Connect to Your Instance (p. 18)	Connect to your instance.
Step 4: Explore Your Instance (p. 19)	Connect to an Amazon EC2 Linux instance and use Linux commands to explore your instance to make the volume available for use.
Step 5: Clean Up (p. 21)	Terminate your instance and delete the volume that you created.

Step 1: Sign Up for Amazon EC2

When you create an Amazon Web Services (AWS) account, AWS automatically signs up the account for all AWS services, including Amazon EC2.

With Amazon EC2, you pay only for what you use. New AWS customers can get started with Amazon EC2 for free. For more information, see Amazon EC2 Pricing.

If you have an AWS account already, skip to the next step. If you don't have an AWS account, use the following procedure to create one.

To create an AWS account

1. Go to http://aws.amazon.com and click **Sign Up Now**.
2. Follow the on-screen instructions.
   Part of the sign-up process involves receiving a phone call and entering a PIN using the phone keypad.

Step 2: Launch an Amazon EC2 Instance

Now that you're signed up for AWS, you're ready to start "computing" in the cloud. The first thing you'll do is to launch a Linux instance using the AWS Management Console. An instance is a virtual server in the cloud. You can use Amazon EC2 to set up and configure the operating system and applications that run on your instance.

You can choose to launch one of the following instances:

- An instance within the Free Usage Tier. The Free Usage Tier enables you to launch and use an Amazon EC2 micro instance free for 12 months. For more information about the Free Usage Tier, see the AWS Free Usage Tier product page and Getting Started with AWS Free Usage Tier.
- A regular instance (not within the Free Usage Tier). During this tutorial you'll incur the standard Amazon EC2 usage fees for the instance, until we show you how to terminate it in the last step. The total charges to complete this tutorial are minimal (typically less than a few dollars). For more information about Amazon EC2 usage fees, see Amazon EC2 Pricing.

To launch an instance

1. Sign in to the AWS Management Console and open the Amazon EC2 console at
   https://console.aws.amazon.com/ec2/.

   Use the email address and password that you specified when signing up for AWS.
2. From the Amazon EC2 console dashboard, click **Launch Instance**.

   The **Create a New Instance** page includes these ways to launch an instance:

   - The **Classic Wizard** offers you precise control and advanced settings for configuring your instance.
   - The **Quick Launch Wizard** automatically configures many selections for you, so that you can get started quickly. This tutorial guides you through the Quick Launch Wizard.

3. On the **Create a New Instance** page, click **Quick Launch Wizard**.

4. (Optional) In **Name Your Instance**, enter a name for the instance that has meaning for you. (If you run multiple instances, naming them helps you identify them in the console.)

5. In **Choose a Key Pair**, you can choose from any existing key pairs that you've created, or you can create a new key pair.

   A key pair enables you to connect to a Linux instance through SSH. Therefore, don't select the **None** option. If you launch your instance without a key pair, then you can't connect to it.

   For this example, we'll create a key pair:

   a. Click **Create New**.

   b. Type a name for your key pair and then click **Download**.

   c. Save your private key in a safe place on your computer. Note the location because you'll need the key to connect to your instance.

6. In **Choose a Launch Configuration**, the Quick Launch Wizard displays a list of basic configurations called Amazon Machine Images (AMIs) that you can choose from to launch your instance. An AMI contains everything you need to create a new instance of a server, such as a web server or a database server. In this tutorial, we'll launch an Amazon Linux instance with a 64-bit operating system. If the configuration is marked with a star, this indicates that it's within the Free Usage Tier.

   Important

   If you launch a regular instance, you're billed from the time that you launch the instance until the instance is stopped or terminated, even if it remains idle.

7. Click **Continue** to view and customize the settings for your instance.

8. In **Security Details**, **Security Group**, you see the security group that the wizard selects for you.

   A security group defines firewall rules for your instances. These rules specify which incoming network traffic is delivered to your instance. All other traffic is ignored.

If you're new to Amazon EC2 and haven't set up any security groups yet, AWS defines a default security group for you. The name and description for the group is quicklaunch-*x* where *x* is a number associated with your quicklaunch group. The first security group you create using the Quick Launch Wizard is named quicklaunch-1. You can change the name and description using the **Edit details** button. The group already has basic firewall rules that enable you to connect to the type of instance you choose. For a Linux instance, you connect through SSH on port 22. The quicklaunch-*x* security group automatically allows SSH traffic on port 22.

If you have used Amazon EC2 before, the wizard looks for an existing security group for the type of instance you're creating.

Caution

The quicklaunch-*x* security group authorizes all IP addresses to access your instance over the specified ports (for example, SSH). This is acceptable for the short exercise in this tutorial, but it's unsafe for production environments. In production, you'll authorize only a specific IP address or range of IP addresses to access your instance.

9.   Review your settings, and click **Launch** to launch the instance.

10. A confirmation page lets you know that your instance is launching. Click **Close** to close the confirmation page and return to the Amazon EC2 console.

11. In the **Navigation** pane, click **Instances** to view the status of your instance. It takes a short time for an instance to launch. The instance's status is `pending` while it's launching.

After the instance is launched, its status changes to `running`.

12. (Optional) After your instance is launched, you can view the quicklaunch-*x* security group rules.

a. On the Amazon EC2 console, in **Network and Security**, click **Security Groups**.

b. Click the quicklaunch-1 security group to view the security rules created by the Quick Launch Wizard.
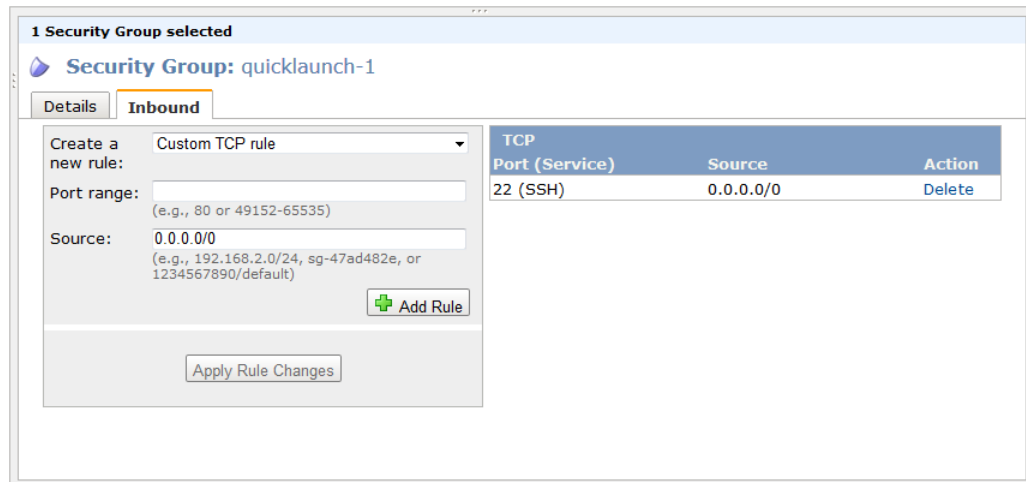
The security group contains a rule that authorizes SSH traffic from any IP address source to port 22. If you launch a Linux instance running Apache and MySQL, the Quick Launch Wizard creates a security group that authorizes traffic to port 80 for HTTP (for web traffic) and port 3306 (for MySQL).

Step 3: Connect to Your Instance

There are several ways to connect to a Linux instance. In this step, we show you how to connect to your instance using your browser. Alternatively, there are tools you can use to connect to your instance instead of connecting using your browser. For more information, see the following topics: Connect to Linux/UNIX Instances from Linux/UNIX with SSH (p. 265) and Connecting to Linux/UNIX Instances from Windows Using PuTTY (p. 269).

To connect to your instance through a web browser

1. You must have Java installed and enabled in the browser. If you don't have Java already, you can contact your system administrator to get it installed, or follow the steps outlined in these pages:

   1. Install Java
   2. Enable Java in your web browser

2. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

3. In the **Navigation** pane, click **Instances**.

4. Right-click your instance, and then click **Connect**.

5. Click **Connect from your browser using the Java SSH client (Java Required)**. AWS automatically detects the public DNS address of your instance and the key pair name you launched the instance with.

6. In **User name**, enter the user name to log in to your instance.

> **Note**
>
> For an Amazon Linux instance, the default user name is `ec2-user`. For Ubuntu, the default user name is `ubuntu`. Some AMIs allow you to log in as `root`. So in this example, you may need to change the user name from `ec2-user` to the appropriate user name.

7. The **Key name** field is automatically populated for you.
8. In **Private key path**, enter the fully qualified path to your .pem private key file.
9. Click **Save key location**, and then click **Stored in browser cache** to store the key location in your browser cache. This ensures that the key location will be detected in subsequent browser sessions, until your clear your browser's cache.
10. Click **Launch SSH Client**.

11. When prompted to add the host to your set of known hosts, click **No**.
12. If necessary, click **Yes** to trust the certificate.
13. Click **Run** to run the MindTerm client.
14. If you accept the license agreement, click **Accept**.
15. If this is your first time running MindTerm, a series of dialog boxes ask you to confirm setup for your home directory and other settings.
16. Confirm settings for MindTerm setup. A screen opens and you are connected to your instance.

Congratulations! You've successfully launched and connected to a Linux instance. If you'd like to explore the instance, continue to the next step. Otherwise, skip to Step 5: Clean Up (p. 21) to terminate the instance so that you don't continue to incur charges.

Step 4: Explore Your Instance

Now that you've connected to your instance, you can use it in the way that you'd use a Linux computer sitting in front of you.

Let's take a look at the default storage for the instance. In MindTerm, run the following command to view the mounted volumes.

```
$ df -h
```

For a micro instance, your output should look something like this.

```
Filesystem             Size   Used   Avail Use% Mounted on
/dev/xvda1             8.0G   1.1G   6.9G  14% /
tmpfs                  298M      0   298M   0% /dev/shm
```

The /dev/xvda1 volume is the root device volume. It contains the image used to boot the instance.

Notice that there's room to install additional software on your instance. For example, you can use **yum** to download and install packages on your instance.

If you need additional storage for data, a simple solution is to add Amazon Elastic Block Store (Amazon EBS) volumes to your Amazon EC2 instance. An Amazon EBS volume serves as network-attached storage for your instance. Let's add a volume to the instance that you've launched. First we'll use the EC2 console to create an Amazon EBS volume and attach it to the instance, and then we'll connect to the instance and mount the volume.

To create and attach an Amazon EBS volume

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

2. In the **Navigation** pane, click **Instances**.

   The console displays the list of current instances. Select your Linux instance. In the **Description** tab in the bottom pane note the **Zone** for the instance.

3. In the **Navigation** pane, in **Elastic Block Store**, click **Volumes**.

   The console displays a list of current EBS volumes. (You should see the EBS volume that serves as the root device volume for your instance.)

4. Click **Create Volume**.

5. In the **Create Volume** dialog box, configure the following settings:

   • Leave the volume size blank. (We'll let the snapshot we select determine the size of the volume.)

   • Select the same **Availability Zone** that you used when you created your instance. Otherwise, you can't attach the volume to your instance.

   • Select a snapshot that contains a public data set hosted by AWS. This enables us to quickly and easily create a volume that is formatted and contains data. The Free Usage Tier provides up to 30 GB of Amazon Elastic Block Storage; therefore, select a public data set that is smaller than 30 GB. We will use `snap-2767d046 -- This is a 10,000 song...`, which contains the Million Song Sample Dataset. This data set is 5 GB in size.

   • Select the **Standard** volume type. This creates a standard EBS volume.

   Now click **Yes, Create**. This creates a new volume (its state is `available`, and its snapshot is `snap-2767d046`).

6. Right-click the newly created volume and select **Attach Volume**.

7. In the **Attach Volume** dialog box, click the following settings:

   • Select your Linux instance from the list.

   • Specify the device name /dev/sdf.

Now click **Yes, Attach**. You'll notice in the **Details** pane for your volume that the volume state is `in-use` and that it is attached to your instance with the device name /dev/sdf. However, if you return to MindTerm and run the **df -h** command again, you won't see the volume yet. That's because we need to mount the volume to make it available. To mount a volume based on a snapshot, run the following commands.

```
$ sudo mkdir /mnt/song-data
$ sudo mount /dev/sdf /mnt/song-data
```

> **Note**
>
> If you didn't specify a snapshot when you created the volume, the new volume is an empty volume. Before you can mount an empty volume that you create, you must format it using the **sudo mkfs** command. For more information, see Making an Amazon EBS Volume Available for Use (p. 455).

Now when you run the **df -h** command, you'll see output like the following.

```
Filesystem            Size   Used  Avail Use% Mounted on
/dev/xvda1            8.6G   1.2G   7.4G  14% /
tmpfs                313M      0   313M   0% /dev/shm
/dev/xvdf            5.0G   4.3G   442M  91% /mnt/song-data
```

To view the contents of the new volume, run the following command.

```
$ dir  /mnt/song-data
```

The output shows the following directory structure.

```
AdditionalFiles  data  LICENSE  lost+found  README
```

At this point, you should have a basic understanding of instances and how you work with them.

When you're finished with your instance, don't forget to clean up any resources you've used and terminate the instance, as shown in the next step.

Step 5: Clean Up

Now that you've completed this tutorial, you can customize the instance to meet your needs and keep using it.

> **Important**
>
> Remember, if you launched an instance in the Free Usage Tier, there are no charges. However, if you launched a regular instance, as soon as your instance starts to boot, you're billed for each hour or partial hour that you keep the instance running, even if the instance is idle. You'll stop incurring charges for a regular instance as soon as the instance status changes to `shutting down` or `terminated`.

When you've decided that you no longer need the instance, you must terminate it. Terminating an instance effectively deletes it. You can't reconnect to the instance after you've terminated it. This differs from stopping the instance; you are still charged for a stopped instance, and you can restart a stopped instance.

To terminate the instance

1. Sign in to the AWS Management Console and open the Amazon EC2 console at
    https://console.aws.amazon.com/ec2/.
2. Locate your instance in the list of instances on the **Instances** page.
3. Right-click the instance, and then click **Terminate**.
4. Click **Yes, Terminate** when prompted for confirmation.
    Amazon EC2 begins terminating the instance.

Amazon EBS volumes can persist even after your instance goes away. If you created and attached an
EBS volume in the previous step, it was detached when you terminated the instance. However, you must
delete the volume, or you'll be charged for volume storage if the storage amount exceeds the limit of the
Free Usage Tier. After you delete a volume, its data is gone and the volume can't be attached to any
instance.

To delete the volume

1. Sign in to the AWS Management Console and open the Amazon EC2 console at
    https://console.aws.amazon.com/ec2/.
2. Locate your volume in the list of volumes on the **Volumes** page.
3. Right-click the volume, and then click **Delete**.
4. Click **Yes, Delete** when prompted for confirmation.
    Amazon EC2 begins deleting the volume.

Finding Additional Tutorials

This tutorial is focused on learning to use Amazon EC2. For tutorials that show you how to use additional
AWS products and services with Amazon EC2, see Getting Started with AWS Documentation.

Amazon Machine Images (AMI)

Topics

Amazon Machine Images (AMI) are the basic building blocks of Amazon EC2. An AMI is a template that contains a software configuration (operating system, application server, and applications) that you can run on Amazon's proven computing environment. Before you can accomplish anything with Amazon EC2, you must understand the concepts in this section.

Related Topics

AMI Basics

An AMI contains all information necessary to boot an Amazon EC2 instance with your software. An AMI is like a template of a computer's root volume. For example, an AMI might contain the software to act as a web server (Linux, Apache, and your web site) or it might contain the software to act as a Hadoop node (Linux, Hadoop, and a custom application). You launch one or more instances from an AMI. An instance might be one web server within a web server cluster or one Hadoop node.

Topics

Available AMIs

Public AMIs can be launched by all AWS accounts. They are advertised and shared in the AWS AMI Catalog. Amazon and the Amazon EC2 community provide a large selection of public AMIs. For more information, go to Amazon Machine Images (AMIs) on the AWS web site.

Paid AMIs are AMIs that you purchase from AWS Marketplace, developers, or through a service contract from an organization such as Red Hat. If you are interested in AWS Marketplace, visit AWS Marketplace. If you're interested in selling an AMI to other developers, go to http://aws.amazon.com/devpay to learn about Amazon DevPay.

The AWS Management Console (at http://aws.amazon.com/console) enables you to search for an AMI that meets specific criteria, and then launch instances from that AMI. For example, you can view the AMIs that Amazon has provided, the AMIs that the EC2 community has provided, or AMIs that use a certain platform (such as Windows, Red Hat, CentOS, or Ubuntu).

You might find that public AMIs suit your needs. However, you might want to customize a public AMI and save that customized AMI for your own use later on. For more information about creating your own AMIs, see Creating Your Own AMIs (p. 35).

After you create a new AMI, you can keep it private so that only you can use it, or you can share it with only specific AWS accounts that you specify. Another option is to make your customized AMI public so that the EC2 community can use it. Building safe, secure, usable AMIs for public consumption is a fairly straightforward process, if you follow a few simple guidelines. For information on how to use shared AMIs and how to share AMIs, see Shared AMIs (p. 29) and Sharing AMIs Safely (p. 62).

To help categorize and manage your AMIs, you can assign *tags* of your choice to them. For more information, see Tagging Your Resources (p. 508).

Storage for the Root Device

All AMIs are categorized as either *backed by Amazon EBS* or *backed by instance store*. The former means that the root device for an instance launched from the AMI is an Amazon EBS volume created from an Amazon EBS snapshot. The latter means that the root device for an instance launched from the AMI is an instance store volume created from a template stored in Amazon S3. For more information, see Root Device Volume (p. 117).

This section summarizes the important differences between the two types of AMIs. The following table provides a quick summary of these differences.

Characteristic	Amazon EBS-Backed	Amazon instance store-backed
Boot Time	Usually less than 1 minute	Usually less than 5 minutes
Size Limit	1 TiB	10 GiB
Root Device Volume	Amazon EBS volume	Instance store volume
Data Persistence	Data on Amazon EBS volumes persists after instance termination; you can also attach instance store volumes that don't persist after instance termination	Data on instance store volumes persists only during the life of the instance; you can also attach Amazon EBS volumes that persist after instance termination
Upgrading	The instance type, kernel, RAM disk, and user data can be changed while the instance is stopped.	Instance attributes are fixed for the life of an instance

Characteristic	Amazon EBS-Backed	Amazon instance store-backed
Charges	Instance usage, Amazon EBS volume usage, and Amazon EBS snapshot charges for AMI storage	Instance usage and Amazon S3 charges for AMI storage
AMI Creation/Bundling	Uses a single command/call	Requires installation and use of AMI tools
Stopped State	Can be placed in stopped state where instance is not running, but the instance is persisted in Amazon EBS	Cannot be in stopped state; instances are running or terminated

Size Limit

Amazon EC2 instance store-backed AMIs are limited to 10 GiB storage for the root device, whereas Amazon EBS-backed AMIs are limited to 1 TiB. Many Windows AMIs come close to the 10 GiB limit, so you'll find that Windows AMIs are often backed by an Amazon EBS volume.

> **Note**
>
> All Windows Server 2008 and Windows Server 2008 R2 AMIs are backed by an Amazon EBS volume by default because of their larger size.

Stopped State

You can *stop* an Amazon EBS-backed instance, but not an Amazon EC2 instance store-backed instance. Stopping causes the instance to stop running (its status goes from *running* to *stopping* to *stopped*). A stopped instance persists in Amazon EBS, which allows it to be restarted. Stopping is different from terminating; you can't restart a terminated instance. Because Amazon EC2 instance store-backed AMIs can't be stopped, they're either running or terminated. For more information about what happens and what you can do while an instance is stopped, see Stopping Instances (p. 310).

Default Data Storage and Persistence

Instances that use instance store for the root device automatically have instance stores available on them (for the root partition and other data you want to add). Any data on those stores is deleted if the instance fails or terminates (except for data on the root device). However, after you launch an instance, you can provide it with persistent non-root data by attaching one or more Amazon EBS volumes.

Instances that use Amazon EBS for the root device automatically have an Amazon EBS volume attached. The volume appears in your list of volumes like any other. The instances don't expose ephemeral storage by default. However, you can add ephemeral storage by using a block device mapping. You can also add additional Amazon EBS volumes for non-root data (for more information, see Block Device Mapping (p. 488)). For information about what happens to the ephemeral storage and volumes when you stop an instance, see Stopping Instances (p. 310).

Boot Times

Amazon EBS-backed AMIs launch faster than Amazon EC2 instance store-backed AMIs. When you launch an Amazon EC2 instance store-backed AMI, all the parts have to be retrieved from Amazon S3 before the instance is available. With an Amazon EBS-backed AMI, only the parts required to boot the instance need to be retrieved from the snapshot before the instance is available. However, the performance of an instance that uses an Amazon EBS volume for its root device is slower for a short time while the

remaining parts are retrieved from the snapshot and loaded into the volume. When you stop and restart the instance, it launches quickly, because the state is stored in an Amazon EBS volume.

AMI Creation

To create Linux/UNIX AMIs backed by instance store, you must create an image of your instance on the instance itself, and there aren't any APIs available to help you. To create a Windows AMI backed by instance store, there's an API call that creates an image, but you still have to call another API to register the AMI.

AMI creation is much easier for AMIs backed by Amazon EBS. There's a single API action called `CreateImage` that works for both Linux/UNIX and Windows. The single call bundles your Amazon EBS-backed AMI and registers it. There's also a button in the AWS Management Console that lets you create an image from a running instance. For more information, see Creating Amazon EBS-Backed AMIs (p. 36).

How You're Charged

With AMIs backed by instance store, you're charged for AMI storage and instance usage. With AMIs backed by Amazon EBS, you're charged for volume storage and usage in addition to the AMI and instance usage charges.

With Amazon EC2 instance store-backed AMIs, each time you customize an AMI and create a new one, all of the parts are stored in Amazon S3 for each AMI. So, the storage footprint for each customized AMI is the full size of the AMI. For Amazon EBS-backed AMIs, each time you customize an AMI and create a new one, only the changes are stored. So the storage footprint for subsequent AMIs you customize after the first is much smaller, resulting in lower AMI storage charges.

When an Amazon EBS-backed instance is stopped, you're not charged for instance usage; however, you're still charged for volume storage. We charge a full instance hour for every transition from a stopped state to a running state, even if you transition the instance multiple times within a single hour. For example, let's say the hourly instance charge for your instance is $0.10. If you were to run that instance for one hour without stopping it, you would be charged $0.10. If you stopped and restarted that instance twice during that hour, you would be charged $0.30 for that hour of usage (the initial $0.10, plus 2 x $0.10 for each restart).

Finding a Suitable AMI

This topic describes how to find an AMI that meets your needs.

AWS Management Console

To find a suitable AMI

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Click **AMIs** in the **Navigation** pane.
   The console displays your AMIs and all public AMIs.
3. To reduce the number of displayed AMIs, select an option from the **Viewing** list boxes to filter the list to the types of AMIs that interest you. For example, select **Amazon Images** to display only Amazon's public images.
4. As you are selecting an AMI, it's important to note whether the AMI is backed by instance store or by Amazon EBS. To verify the type of root device volume that is used by an AMI, select the AMI. In the lower pane of the EC2 console, check the value of **Root Device Type** (`ebs` or `instance store`).

For more information about the differences between these AMI types, see Storage for the Root Device (p. 24).

5. After locating an AMI that meets your needs, write down its AMI ID, which has the form ami-*xxxxxxxx*. You can use this ID to launch instances of the AMI or register your own AMI, using this one as a baseline.

Command Line Tools

To find a suitable AMI

1. Use the ec2-describe-images command to list your AMIs and Amazon's public AMIs.

```
PROMPT> ec2-describe-images -o self -o amazon
```

The following example shows only part of the resulting output from the command (information for 10 AMIs).

```
IMAGE ami-d8699bb1 amazon/ami-vpc-nat-1.0.0-beta.i386-ebs amazon available
 public  i386 machine aki-407d9529   ebs paravirtual xen
BLOCKDEVICEMAPPING /dev/sda1  snap-33d88c5f 8
IMAGE ami-c6699baf amazon/ami-vpc-nat-1.0.0-beta.x86_64-ebs amazon available
 public  x86_64 machine aki-427d952b   ebs paravirtual xen
BLOCKDEVICEMAPPING /dev/sda1  snap-57d88c3b 8
IMAGE ami-30f30659 amazon/amzn-ami-0.9.7-beta.i386-ebs amazon available
public  i386 machine aki-407d9529   ebs paravirtual xen
BLOCKDEVICEMAPPING /dev/sda1  snap-d895cdb3 10
IMAGE ami-0af30663 amazon/amzn-ami-0.9.7-beta.x86_64-ebs amazon available
public  x86_64 machine aki-427d952b   ebs paravirtual xen
BLOCKDEVICEMAPPING /dev/sda1  snap-f295cd99 10
IMAGE ami-3ac33653 amazon/amzn-ami-0.9.8-beta.i386-ebs amazon available
public  i386 machine aki-407d9529   ebs paravirtual xen
BLOCKDEVICEMAPPING /dev/sda1  snap-14ba967f 10
IMAGE ami-38c33651 amazon/amzn-ami-0.9.8-beta.x86_64-ebs amazon available
public  x86_64 machine aki-427d952b   ebs paravirtual xen
BLOCKDEVICEMAPPING /dev/sda1  snap-10b9957b 10
IMAGE ami-08728661 amazon/amzn-ami-0.9.9-beta.i386-ebs amazon available
public  i386 machine aki-407d9529   ebs paravirtual xen
BLOCKDEVICEMAPPING /dev/sda1  snap-674a930d 10
IMAGE ami-2272864b amazon/amzn-ami-0.9.9-beta.x86_64-ebs amazon available
public  x86_64 machine aki-427d952b   ebs paravirtual xen
BLOCKDEVICEMAPPING /dev/sda1  snap-8926ffe3 10
IMAGE ami-76f0061f amazon/amzn-ami-2010.11.1-beta.i386-ebs amazon available
 public  i386 machine aki-407d9529   ebs paravirtual xen
BLOCKDEVICEMAPPING /dev/sda1  snap-cba692a1 8
IMAGE ami-74f0061d amazon/amzn-ami-2010.11.1-beta.x86_64-ebs amazon available
 public  x86_64 machine aki-427d952b   ebs paravirtual xen
BLOCKDEVICEMAPPING /dev/sda1  snap-ffa69295 8
IMAGE ami-8c1fece5 amazon/amzn-ami-2011.02.1.i386-ebs amazon available public
  i386 machine aki-407d9529   ebs paravirtual xen
BLOCKDEVICEMAPPING /dev/sda1  snap-22fc264e 8
IMAGE ami-8e1fece7 amazon/amzn-ami-2011.02.1.x86_64-ebs amazon available
public  x86_64 machine aki-427d952b   ebs paravirtual xen
BLOCKDEVICEMAPPING /dev/sda1  snap-a6fc26ca 8
```

2. To reduce the number of displayed AMIs, use a filter to list only the types of AMIs that interest you. For example, use `--filter "platform=windows"` to display only Windows-based AMIs.

3. After locating an AMI that meets your needs, write down its AMI ID, which has the form ami-*xxxxxxxx*. You can use this ID to launch instances of the AMI or register your own AMI, using this one as a baseline.

API

To find a suitable AMI

1. Use the DescribeImages action to to list all Amazon s. Construct the following request.

```
https://ec2.amazonaws.com/
?Action=DescribeImages
&User.1=amazon
&AUTHPARAMS
```

The following is an example response.

```
<DescribeImagesResponse xmlns="http://ec2.amazonaws.com/doc/2012-07-20/">
  <imagesSet>
      <item>
        <imageId>ami-8c1fece5</imageId>
       <imageLocation>amazon/amzn-ami-2011.02.1.i386-ebs</imageLocation>

        <imageState>available</imageState>
        <imageOwnerId>137112412989</imageOwnerId>
        <isPublic>true</isPublic>
        <architecture>i386</architecture>
        <imageType>machine</imageType>
        <kernelId>aki-407d9529</kernelId>
        <imageOwnerAlias>amazon</imageOwnerAlias>
        <name>amzn-ami-2011.02.1.i386-ebs</name>
        <description>Amazon Linux AMI i386 EBS</description>
        <rootDeviceType>ebs</rootDeviceType>
        <rootDeviceName>/dev/sda1</rootDeviceName>
        <blockDeviceMapping>
          <item>
            <deviceName>/dev/sda1</deviceName>
            <ebs>
               <snapshotId>snap-22fc264e</snapshotId>
               <volumeSize>8</volumeSize>
               <deleteOnTermination>true</deleteOnTermination>
            </ebs>
          </item>
        </blockDeviceMapping>
        <virtualizationType>paravirtual</virtualizationType>
        <hypervisor>xen</hypervisor>
      </item>
  </imagesSet>
```

2. To reduce the number of displayed AMIs, use a filter to list only the types of AMIs that interest you. For example, use the following to display only Windows-based AMIs.

```
&Filter.1.Name=platform
&Filter.1.Value.1=windows
```

3. After locating an AMI that meets your needs, write down its AMI ID, which has the form ami-*xxxxxxxx*. You can use this ID to launch instances of the AMI or register your own AMI, using this one as a baseline.

Shared AMIs

This topic 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.

Topics
- Find Shared AMIs (p. 29)
- Safe Use of Shared AMIs (p. 30)

Find Shared AMIs

You can find a shared AMI using the console, the command line tools, or the API.

AWS Management Console

To find a shared AMI

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Click **AMIs** in the **Navigation** pane.

   The console displays your AMIs and all public AMIs.

3. To reduce the number of displayed AMIs, select an option from the **Viewing** list boxes to filter the list to the types of AMIs that interest you. For example, select **Amazon Images** to display only Amazon's public images.

Command Line Tools

To find shared AMIs, use the ec2-describe-images command with a flag to filter the results.

The following command displays a list of all public AMIs. The `-x all` flag shows AMIs that can be used by all AWS accounts to launch an instance (in other words, AMIs with public launch permissions). This includes the AMIs that you own with public launch permissions.

```
PROMPT> ec2-describe-images -x all
```

The following command displays a list of AMIs for which you have explicit launch permissions. Any such AMIs that you own are excluded from the list.

```
PROMPT> ec2-describe-images -x self
```

The following command displays a list of AMIs owned by Amazon.

```
PROMPT> ec2-describe-images -o amazon
```

The following command displays a list of AMIs owned by a particular AWS account.

```
PROMPT> ec2-describe-images -o <target_uid>
```

The `<target_uid>` is the account ID that owns the AMIs you're looking for.

To reduce the number of displayed AMIs, use a filter to list only the types of AMIs that interest you. For example, use `--filter "platform=windows"` to display only Windows-based AMIs.

API

To find a shared AMI

1. Use the DescribeImages action to to list all Amazon s. Construct the following request.

    ```
    https://ec2.amazonaws.com/
    ?Action=DescribeImages
    &User.1=amazon
    &AUTHPARAMS
    ```

2. To reduce the number of displayed AMIs, use a filter to list only the types of AMIs that interest you.
    For example, use the following to display only Windows-based AMIs.

    ```
    &Filter.1.Name=platform
    &Filter.1.Value.1=windows
    ```

Safe Use of Shared AMIs

You launch AMIs at your own risk. Amazon cannot vouch for the integrity or security of AMIs shared by other EC2 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 web site, another EC2 user, etc). If you do not know the source of an AMI, we recommend 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 plan to use a shared AMI, review the following table to confirm the instance is not doing anything malicious.

Launch Confirmation Process

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 (p. 63).
4	Check if SSH allows root password logins. See Disable Password-Based Logins for Root (p. 63) 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.

For more information, see Sharing AMIs Safely (p. 62).

Paid AMIs

Topics
- Find Paid AMIs (p. 31)
- Purchase a Paid AMI (p. 32)
- Launch a Paid AMIs (p. 32)
- Using Paid Support (p. 33)
- Bills for Paid and Supported AMIs (p. 33)

Amazon EC2 integrates with Amazon DevPay, enabling developers to charge other EC2 users for the use of their AMIs or to provide support for instances. To learn more about Amazon DevPay go to the Amazon DevPay Developer Guide. For more information about charging for the use of your AMIs, or providing support, see Creating Paid AMIs (p. 53).

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.

Find Paid AMIs

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

You can tell if an AMI is a paid AMI by describing the image using the console, the command line tools, or the API. If the AMI is a paid AMI, it has a product code. Otherwise, it does not.

Note

You must sign up for a paid AMI before you can launch it. If you find a paid AMI that you are interested in, go to the Amazon EC2 resource center and forums, which might have more information about the paid AMI and where you can sign up to use it.

AWS Management Console

To find a paid AMI

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Click **AMIs** in the **Navigation** pane.
   The console displays your AMIs and all public AMIs.
3. Select **AWS Marketplace** from the **Viewing** list box to filter the list to the paid AMIs.

Command Line Tools

To check whether an AMI is a paid AMI

1. Use the ec2-describe-images command as follows.

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

   The <ami_id> is the ID of the AMI.

   The command returns numerous fields that describe the AMI. If the output for the AMI contains a product code, the AMI is a paid AMI.
2. This example shows **ec2-describe-images** output for a paid AMI. The product code is ACD42B6F.

   ```
   PROMPT> ec2-describe-images ami-a5bf59cc
   IMAGE   ami-a5bf59cc    cloudmin-2.6-paid/image.manifest.xml    541491349868
       available public   ACD42B6F        i386    machine   instance-store
   ```

Purchase 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 are taken to a page where you see the price for the paid AMI's and can confirm that you want to purchase the AMI.

Important

You don't get the discount from Amazon EC2 Reserved Instances with paid AMIs. That is, if you purchase Reserved Instances, you don't get the lower price associated with them when you launch a paid AMI. You always pay the price that the seller of the paid AMI specified. For more information about Reserved Instances, see Reserved Instances.

Launch a Paid AMIs

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. For more information, see Launching Amazon EC2 Instances (p. 217).

The instance is charged according to the rates set by the owner of the AMI (which is more than the base Amazon EC2 rate).

Note

The owner of a paid AMI can confirm if a particular instance was launched using their paid AMI.

Using Paid Support

The paid AMI feature also enables 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 enables 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.

Important

After you set the product code attribute, it cannot be changed or removed.

To associate a product code with your AMI, use the ec2-modify-image-attribute command:

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

The *<ami_id>* is the AMI ID and *<product_code>* is the product code.

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
```

Important

If you've purchased Amazon EC2 Reserved Instances, you can't use them with supported AMIs. That is, if you associate a product code with one of your AMIs, you don't get the lower price associated with your Reserved Instances when you launch that AMI. You always pay the price that the seller of the support product specified. For more information about Reserved Instances, see Reserved Instances.

Bills for Paid and Supported AMIs

At the end of each month, you receive an email 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 (go to http://www.amazon.com/dp-applications).

Creating Amazon EBS-Backed AMIs Using the Console

Topics

This section walks you through the process of creating an Amazon EBS-backed AMI from a running Amazon EBS-backed instance. For instructions that use the command line tools or API, see Creating Amazon EBS-Backed AMIs (p. 36).

Create an AMI from an Instance

To create an AMI from a running Amazon EBS-backed instance

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. If you don't have a running instance that uses an Amazon EBS volume for the root device, you must launch one. For instructions, see Launching an Instance from an AMI (p. 217).
3. [Optional] Connect to the instance and customize it however you want. For example, you can install software and applications, copy data, or attach additional EBS volumes. For more information about connecting to an instance, see Connecting to Amazon EC2 Instances (p. 255).
4. In the **Navigation** pane, click **Instances** to view a list of your instances. Right-click your running instance and select **Create Image (EBS AMI)**.

    Tip

    If this option is disabled, your instance isn't an Amazon EBS-backed instance.

    The **Create Image** dialog box appears.

5. Fill in the requested information as follows, and click **Yes, Create**.

    a. A unique name for the image.
    b. [Optional] A description of the image (up to 255 characters).
    c. By default, Amazon EC2 shuts down the instance, takes snapshots of any attached volumes, creates and registers the AMI, and then reboots the instance. Select **No Reboot** if you don't want your instance to be shut down.

        Warning

        If you select the **No Reboot** option, the file system integrity of the created image can't be guaranteed.

    d. [Optional] You can modify the root volume, EBS volumes, and instance store volumes as follows:

        *   To change the size of the root volume, click **Root Volume**, fill in the **Volume Size** field, and click **Save**.
        *   To suppress an EBS volume specified by the block device mapping of the AMI used to launch the instance, click **EBS Volumes**, go to the volume, and click **Delete**.
        *   To add an EBS volume, click **EBS Volumes**, fill in the fields (**Device**, **Snapshot**, and so on), and click **Add**. When you then launch an instance from your new AMI, these additional volumes are automatically attached to the instance. Empty volumes must be formatted and mounted. Volumes based on a snapshot must be mounted.
        *   To suppress an instance store volume specified by the block device mapping of the AMI used to launch the instance, click **Instance Store Volumes**, go to the volume, and click **Delete**.
        *   To add an instance store volume, click **Instance Store Volumes**, select the **Instance Store** and **Device**, and click **Add**. When you launch an instance from your new AMI, these additional volumes are automatically initialized and mounted. These volumes don't contain data from the instance store volumes of the running instance from which you based your AMI.

        For more information, see Root Device Volume (p. 117), Amazon EC2 Instance Store (p. 479), and Block Device Mapping (p. 488)

6. In the **Navigation** pane, select **AMIs** page to view the AMI's status. While the new AMI is being created, its status is `pending`.

AMI ID	Source	Owner	Visibility	Status	Platform	Root Device Type
ami-634ba70a	N/A	xxxxxxxxxxxx	Private	○ pending	△ Other Linux	ebs

It takes a few minutes for the whole process to finish.

7. After your new AMI's status is `available`, go to the **Snapshots** page and view the new snapshot that was created for the new AMI. Any instance you launch from the new AMI uses this snapshot for its root device volume.

Snapshot ID	Capacity	Description	Status	Started ▼	Progress
snap-1981a370	15 GiB	Created by CreateImage(i-a3db28c8)	● completed	2010-02-22 18:31 PST	available (100%)

8. Go back to the **AMIs** page, right-click the image, and select **Launch Instance**.
The launch wizard opens.

9. Walk through the wizard to launch an instance of your new AMI.

10. After your instance's status is `running`, connect to the instance and verify that any changes you made to the original AMI have persisted.

You now have a new AMI and snapshot that you just created. Both continue to incur charges to your account until you stop or delete them.

Delete an AMI and a Snapshot

To delete an AMI and a snapshot

1. Go to the **AMIs** page. Right-click the AMI and select **De-register Image** When asked for confirmation, click **Yes, De-register**.
The image is de-registered, which means it is deleted and can no longer be launched.

2. Go to the **Snapshots** page. Right-click the snapshot and select **Delete Snapshot**. When asked for confirmation, click **Yes, Delete**.
The snapshot is deleted.

Creating Your Own AMIs

There are many public AMIs available to you. To see the available AMIs, go to Amazon Machine Images (AMIs). If the available public AMIs don't provide everything that you're looking for, you can create an AMI that meets your needs.

Creating your own AMI helps you make the most of Amazon EC2. Your AMI becomes the basic unit of deployment; it enables you to rapidly boot new custom instances as you need them. This section gives an overview of your AMI creation options, identifies the tools you need, and walks you through the process.

Before you begin this section, you should be familiar with AMI and instance concepts. For more information, see the following sections:

- Amazon Elastic Block Store (p. 434)

Overview of the AMI Creation Process

There are a few different ways to create an AMI. The process you must follow to create an AMI depends on whether you are creating an Amazon EBS-backed AMI or an Amazon EC2 instance store-backed AMI. There are significant differences between Amazon EBS-backed and Amazon EC2 instance store-backed AMIs, such as data persistence. For information on the differences between these choices, see Storage for the Root Device (p. 24).

First, decide which operating system and root device volume you want, and then you'll know which of these processes to use for creating the AMI:

- **Amazon EBS-backed AMI**
  The same general process applies to Linux/UNIX and Windows.
  - **Linux/UNIX—**Creating Amazon EBS-Backed AMIs (p. 36)
  - **Windows—** Creating an Amazon EBS-Backed Windows AMI (Windows User Guide)
- **Amazon EC2 instance store-backed AMI**
  The process depends on the operating system.
  - **Linux/UNIX—**Creating Instance Store-Backed Linux/UNIX AMIs (p. 39)
  - **Windows—** Creating an Instance Store-Backed Windows AMI (Windows User Guide)

Creating Amazon EBS-Backed AMIs

Topics
- Process for Creating Amazon EBS-Backed AMIs (p. 37)
- Special Cases (p. 37)
- How to Create Amazon EBS-Backed AMIs (p. 38)
- Converting Amazon EC2 instance store-backed AMIs to EBS-Backed AMIs (p. 39)

The process for creating an Amazon EBS-backed AMI is simple. You start with an Amazon EBS-backed AMI (for example, one of the public AMIs that Amazon provides), and modify it to suit your particular needs. You can't create an Amazon EBS-backed AMI using these instructions if you start with an Amazon EC2 instance store-backed instance. For instructions using an instance store-backed instance, see Creating Instance Store-Backed Linux/UNIX AMIs (p. 39).

If the AMI you start with doesn't already have the storage devices you want attached, you can add them by creating EBS volumes or using block device mapping. To create EBS volumes, use `ec2-create-volume` and `ec2-attach-volume` or `CreateVolume` and `AttachVolume`. You also can call `ec2-run-instances` or `RunInstances` with block device mapping information for the devices you want to add. For more information about block device mapping, see Block Device Mapping (p. 488).

When you have a running instance in the state you want, use **ec2-create-image** or `CreateImage` and specify the instance ID. Amazon EC2 powers down the instance, takes images of any volumes that were attached, creates and registers the AMI, and then reboots the instance. It takes several minutes for the entire process to complete. If you customized the instance with instance store volumes or additional EBS volumes besides the root device, the new AMI contains block device mapping information for those volumes. When you launch an instance from your new AMI, the instance automatically launches with the additional volumes. The instance store volumes are new and don't contain any data from the instance store volumes of the original instance used to create the AMI.

The reason Amazon EC2 powers down the instance before creating the AMI is to ensure that everything on the instance is stopped and in a consistent state during the creation process. If you're confident that your instance is in a consistent state appropriate for AMI creation, you can add the --no-reboot flag to **ec2-create-image** or `CreateImage` that tells Amazon EC2 not to power down and reboot the instance. With this flag, the instance remains running throughout the AMI creation process. Some file systems, such as xfs, can freeze and unfreeze activity, making it safe to create the image without rebooting the instance.

Note

Amazon EBS-backed instances are stored as Amazon EBS data. Standard storage rates apply.

Important

You cannot delete a snapshot of the root device of a registered EBS-backed AMI. You must first de-register the AMI before you can delete the snapshot.

Note

You cannot register an image where a secondary (non-root) snapshot has AWS Marketplace product codes.

For an overview of the AWS Marketplace, go to https://aws.amazon.com/marketplace/help/200900000. For details on how to use the AWS Marketplace, see AWS Marketplace.

Process for Creating Amazon EBS-Backed AMIs

The following table describes the process for creating an Amazon EBS-backed AMI. The information is applicable to both Linux/UNIX and Windows AMIs.

How to Create an Amazon EBS-Backed AMI

1	Launch an instance of an Amazon EBS-backed AMI that is similar to the AMI you want to create. For example, you might take a public AMI that uses the operating system you want to use for your AMI. The instance must be launched from an Amazon EBS-backed AMI; you can't use an Amazon EC2 instance store-backed AMI.
2	When the instance is running, customize it as you want. For example, you could attach additional Amazon EBS volumes, load applications, or copy data. **Important** If you customize your instance with instance store volumes or additional EBS volumes besides the root device, the new AMI contains block device mapping information for those volumes. When you then launch an instance from your new AMI, the instance automatically launches with the additional volumes.
3	After the instance is set up the way you want it, create an image from that instance.

Special Cases

In some cases, the general tasks in creating Amazon EBS-backed AMIs don't apply:

- You don't have the original AMI to launch instances from.

In this case, you can create an Amazon EBS-backed AMI by registering a snapshot of a root device. You must own the snapshot and it must be a Linux/UNIX system (this process is not available for Windows instances). For more information about creating an AMI this way, see Launching an Instance from a Snapshot (p. 222).

- You have an Amazon EC2 instance store-backed Linux/UNIX AMI.
  In this case, you can convert the AMI to be backed by Amazon EBS. You cannot convert a Windows AMI backed by instance store. For more information about converting a Linux/UNIX AMI, see Converting Amazon EC2 instance store-backed AMIs to EBS-Backed AMIs (p. 39).

How to Create Amazon EBS-Backed AMIs

You can create an Amazon EBS-backed AMI by using the AWS Management Console, the command line tools, or the API. The following section describes the steps using each tool or interface.

AWS Management Console

For instructions that use the AWS Management Console, see Creating Amazon EBS-Backed AMIs Using the Console (p. 33).

Command Line Tools

To create an Amazon EBS-backed AMI

1. Enter the following command to create an image:

```
PROMPT>  ec2-create-image -n your_image_name  instance_id
```

For example:

```
PROMPT>  ec2-create-image -n "My AMI" i-eb977f82
```

Amazon EC2 creates an image and returns an AMI ID.

```
IMAGE ami-8675309
```

2. If you want to check whether the AMI is ready, enter the following command:

```
$  ec2-describe-images -o self
```

Amazon EC2 returns information about the AMI.

API

To create an Amazon EBS-backed AMI

- Construct the following Query request to create an image:

```
https://ec2.amazonaws.com/
?Action=CreateImage
&InstanceId=instance_id
```

```
&Name=My_Ami
&AUTHPARAMS
```

In the following example response, Amazon EC2 creates the image and returns its AMI ID.

```
<CreateImageResponse xmlns="http://ec2.amazonaws.com/doc/2012-07-20/">
  <imageId>ami-8675309</imageId>
</CreateImageResponse>
```

AMI creation can take time. You can check whether the AMI is ready by using .

Converting Amazon EC2 instance store-backed AMIs to EBS-Backed AMIs

There's no simple API or button in the AWS Management Console that converts an existing Amazon EC2 instance store-backed AMI to an Amazon EBS-backed AMI.

Important

It isn't possible to convert an Amazon EC2 instance store-backed Windows AMI. You need to take a public Amazon EBS-backed Windows AMI, modify it to meet your specifications, and then create an image from it. For information, see Creating Amazon EBS-Backed AMIs Using the Console (p. 33).

You can convert Amazon EC2 instance store-backed Linux/UNIX AMIs to EBS-backed systems. The following table describes the process.

How to convert a Linux/UNIX Amazon EC2 instance store-backed AMI to an EBS-backed AMI

1	Copy the AMI's root device information to an Amazon EBS volume. For more information, see the related task list in Amazon EC2 Root Device Storage Usage Scenarios (p. 119)
2	Create a snapshot of that volume. For more information, see Creating an Amazon EBS Snapshot (p. 465).
3	Register the image with a block device mapping that maps the root device name of your choice to the snapshot you just created. For an example, see Block Device Mapping (p. 488).

Additional Resources

You might find it useful to refer to available blog posts that discuss conversion. The following are two example blogs; AWS, however, takes no responsibility for the completeness or accuracy of the content:

- http://www.elastician.com/2009/12/creating-ebs-backed-ami-from-s3-backed.html
- http://coderslike.us/2009/12/07/amazon-ec2-boot-from-ebs-and-ami-conversion/

Creating Instance Store-Backed Linux/UNIX AMIs

Topics
- Tools You Need (p. 40)

- Creating an Instance Store-Backed AMI From an Existing AMI (p. 41)
- Creating an Instance Store-Backed AMI From a Loopback (p. 45)

For Linux/UNIX systems, you have two common ways to prepare Amazon EC2 instance store-backed AMIs. The easiest method (A) involves launching an existing public AMI and modifying it according to your requirements. For more information, see Creating an Instance Store-Backed AMI From an Existing AMI (p. 41).

Another approach (B) is to build a fresh installation either on a stand-alone machine or on an empty file system mounted by loopback. The process entails building an operating system installation from scratch. After you've built the installation package to your satisfaction, you must bundle it using the AMI tool for bundling volumes and register it using the command line tool for registering images. For information, see Creating an Instance Store-Backed AMI From a Loopback (p. 45).

The following diagram shows the general tasks in creating Amazon EC2 instance store-backed Linux/UNIX AMIs.

Create Amazon S3-backed Linux/UNIX AMIs

(A) Existing AMI

| Customize a Linux/UNIX instance | Upload the key and certificate | Bundle customized image | Upload bundle | Register image |
| 1 | 2 | 3 | 4 | 5 |

Loopback (B)

| Install Linux/UNIX OS on clean root file system | Bundle image of new installation | Upload bundle | Register image |
| 1 | 2 | 3 | 4 |
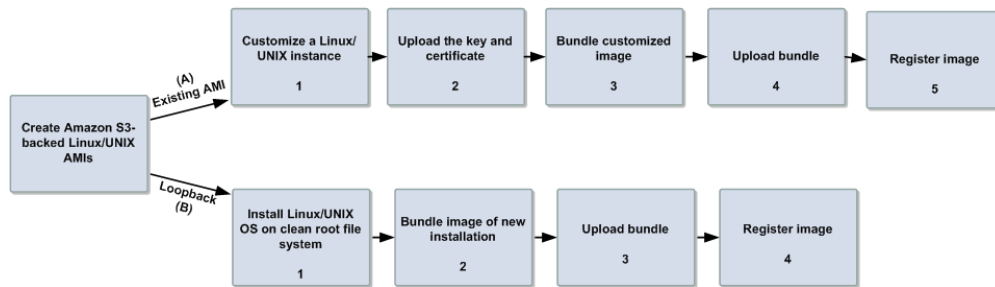
This section discusses the steps for creating AMIs from an existing file and from a loopback, and some basics about the AMI tools.

Tools You Need

Amazon created the Amazon EC2 AMI tools to help you perform specific tasks for Amazon EC2 instance store-backed Linux/UNIX AMIs. You use these AMI tools, which are a set of command line utilities, for bundling and uploading Amazon EC2 instance store-backed Linux/UNIX AMIs. You also use these AMI tools for managing these bundled images. For information about the specific AMI tools, see AMI Tools Reference in the *Amazon Elastic Compute Cloud Command Line Reference*.

When you bundle an Amazon EC2 instance store-backed Linux/UNIX AMI and you start with an instance, you use the AMI tools for bundling and uploading the bundle, and then you use the API tools to register the image. If you are creating an Amazon EC2 instance store-backed AMI via loopback, you first prepare the instance, then use the AMI tools to bundle before you use the API tools to register the image you created.

If you are starting with an instance of an Amazon public AMI, it might already have the AMI tools installed. Try running the command `ec2-bundle-vol` to check if the instance already has the AMI tools.

If the tools are already installed, you can jump to the section that discusses the bundling process you want to complete:

- Creating an Instance Store-Backed AMI From an Existing AMI (p. 41)
- Creating an Instance Store-Backed AMI From a Loopback (p. 45)

If the tools aren't installed, read on. This section describes installation and usage information when using AMI tools.

Install the AMI Tools

The AMI tools are available in both a Zip file and 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.

For information about installing the AMI tools, go to Amazon EC2 AMI Tools.

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
    ```

View the AMI Tools Documentation

This section describes how to view Linux/UNIX documentation.

To view the manual for each tool

*   Append `--manual` to the command that invokes the tool.

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

To view help for each tool

*   Append `--help` to the command that invokes the tool.

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

Creating an Instance Store-Backed AMI From 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.

Before You Get Started

| 1 | Before you select an AMI, determine whether the instance types you plan to launch are 32-bit or 64-bit. For more information, see Instance Families and Types (p. 85). |
| 2 | Make sure you are using GNU Tar 1.15 or later. |

Tasks to Use an Existing AMI to Create a New AMI

Customize an Instance

Customizing an instance involves the following series of steps:

1. Selecting an AMI from available AMIs.
2. Launching an instance from the AMI you selected.
3. Making changes to (thus, *customizing*) the instance, such as altering the Linux configuration, adding software, and configuring web applications.

For more information, see Launching Amazon EC2 Instances (p. 217).

After you've launched an instance according to your specifications, proceed to the next steps to create a new AMI using the customized instance.

Upload the Key and Certificate

Your new AMI should be encrypted and signed to ensure that only you and Amazon EC2 can access it. To make this happen, you must upload your Amazon EC2 private key and X.509 certificate to an instance store directory on your running instance. The private key and the certificate will be used in the AMI bundling process. For information on obtaining your Amazon EC2 private key and X.509 certificate, see How to Create an X.509 Certificate and Private Key (p. 352).

An Amazon Linux AMI mounts the instance store on */media/ephemeral0*. If you are using an Amazon Linux AMI, you must first login to your running instance as *ec2-user* and use the following code to grant write permissions to the instance store before you can upload the private key and the certificate.

```
$  sudo chmod 777 /media/ephemeral0
```

> **Note**
>
> Non Amazon Linux AMIs use different login names (for example, root, ubuntu, etc.,) and a different location to mount the instance store (for example, */mnt*).

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

- Copy your Amazon EC2 private key and X.509 certificate to the directory where an instance store is mounted using a secure copy function such as SCP.

  The following code shows the syntax to use with the `scp` command.

```
$ scp -i <keypair_name> <private_keyfile> <certificate_file> <user
name>@<dns_location>
        <instance store directory>
```

Where,

Parameter	Description
keypair_name	Name of your key pair.
private_keyfile	File that contains the private key.
certificate_file	File that contains the certificate.
username	Login name you use to log in to your instance.
dns_location	DNS location of the instance within Amazon EC2.
instance store directory	Directory where your instance store is mounted.

> **Note**
>
> It is important to upload the key and certificate files into the instance store directory of your instance to prevent them from being bundled with the new AMI.

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

Example

The following is an example of a fully specified `scp` command using the Amazon Linux AMI.

```
$ scp  -i id_rsa-gsg-keypair pk-HKZYKTAIG2ECMXYIBH3HXV4ZBEXAMPLE.pem
cert-HKZYKTAIG2ECMXYIBH3HXV4ZBEXAMPLE.pem
ec2-user@ec2-67-202-51-223.compute-1.amazonaws.com:/media/ephemeral0
pk-HKZYKTAIG2ECMXYIBH3HXV4ZBEXAMPLE.pem  100%  717     0.7KB/s   00:00
cert-HKZYKTAIG2ECMXYIBH3HXV4ZBEXAMPLE.pem
100%  685     0.7KB/s   00:00
```

Bundle a Customized Image (Requires Root Privileges)

When you have the image that meets your specifications, you need to bundle it.

To bundle a customized image

• Enter the following command:

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

The `<private_keyfile>` is the file that contains the private key, `<certificate_file>` is the file that contains the certificate, and `<user_id>` is the ID associated with your AWS account.

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

Note

The user ID is your AWS account ID without dashes. It consists of 12 to 15 characters, and it's *not* the same as your Access Key ID. For information about viewing your account ID, see Viewing Your Account ID (p. 353).

Example

This command bundles the local machine root file system.

```
# ec2-bundle-vol -k pk-HKZYKTAIG2ECMXYIBH3HXV4ZBEXAMPLE.pem -c cert-HKZYK
TAIG2ECMXYIBH3HXV4ZBEXAMPLE.pem -u 111122223333

Please specify a value for arch [i386]:
Copying / into the image file /tmp/image...
Excluding:
/sys
...
...
1+0 records in
1+0 records out
1048576 bytes (1.0 MB) copied, 0.00172 s, 610 MB/s
mke2fs 1.40.4 (31-Dec-2007)
...
Bundling image file...
Splitting /tmp/image.tar.gz.enc...
Created image.part.00
Created image.part.01
...
Created image.part.NN
Generating digests for each part...
Digests generated.
Creating bundle manifest...
ec2-bundle-vol complete.
```

Upload a Bundled AMI

You must upload the bundled AMI to Amazon S3 before Amazon EC2 can access it. This task is necessary when you create Amazon EC2 instance store-backed AMIs from an existing file or from a loopback. 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 AWS account, the `ec2-upload-bundle` command will fail.

To upload the bundled AMI

• Enter the following command:

```
# ec2-upload-bundle -b <bucket> -m <manifest_path> -a <access_key> -s
<secret_key>
```

The `<bucket>` is the target bucket (you can bundle to a "subfolder"; e.g., myawsbucket/ami-image-folder), `<access_key>` is your AWS Access Key, and `<secret_key>` is

your AWS Secret Key. The `-m <manifest_path>` is the full path to the manifest file (e.g., /tmp/image.manifest.xml).

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.

Register the AMI

Your must register your image with Amazon EC2, so Amazon EC2 can locate it and run instances based on it. This task is necessary when you create Amazon EC2 instance store-backed AMIs from an existing file or from a loopback.

Note

If you make any changes to the source image stored in Amazon S3, you must reregister the image.

To register the AMI that you created and uploaded to Amazon S3

- Enter the following command (note that it is part of the Amazon EC2 API tools, and not the AMI tools):

```
PROMPT> ec2-register <your-s3-bucket>/image.manifest.xml -n image_name
IMAGE ami-2bb65342
```

Amazon EC2 returns an AMI identifier, the value next to the `IMAGE` tag (`ami-2bb65342` in the example), that you can use to run instances.

Creating an Instance Store-Backed AMI From a Loopback

Creating AMIs through a loopback 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. After you have installed your operating system, you can bundle the resulting image as an AMI with the `ec2-bundle-image` command, which is part of the AMI tools (and not an API action). For more information about the `ec2-bundle-image` command and the AMI tools, go to the Amazon Elastic Compute Cloud Command Line Reference.

Note

This method works only with AMIs that use instance stores for their root devices. This method is not applicable for AMIs backed by Amazon EBS.

Before You Get Started

1	Before you select an AMI, determine whether the instance types you plan to launch are 32-bit or 64-bit. For more information, see Instance Families and Types (p. 85).
2	Make sure you are using GNU Tar 1.15 or later.
3	This topic uses Fedora Core 4. Please make any adjustments for your distribution.

Tasks to Create a New AMI Through a Loopback

1	Install Linux/UNIX and Prepare the System
	a. Create a File to Host the AMI (p. 46)
	b. Create a Root File System Inside the File (p. 46)
	c. Mount the File through Loopback (p. 47)
	d. Prepare for the Installation (p. 48)
	e. Install the Operating System (p. 49)
	f. Configure the Operating System (p. 50)
2	Bundle the Loopback File Image (p. 51)
3	Upload a Bundled AMI (p. 52)
4	Register the AMI (p. 53)

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/UNIX installation requires about 700 MB, 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 count=size
```

The <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 example creates a 1 GB file (1024*1 MB).

```
# 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

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

To create an ext3 file system

- Enter the following command:

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

The <image_name> is the name of the image file.

Example

The following example 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 enables 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. Enter the following command to create a mount point in the file system where the image will be attached:

    ```
    # mkdir <image_mountpoint>
    ```

    The *<image_mountpoint>* is the location where the image will be mounted.

2. Mount the file system image:

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

    The *<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
```

   The `<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 content.

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

[base]
name=Fedora Core 4 - $basearch - Base
mirrorlist=http://fedora.redhat.com/download/mirrors/fedora-core-$releasever
enabled=1

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

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

4. Enter the following:

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

The *<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 preceding 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 groupin
stall Base
```

The *<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 your 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. Verify that the following line appears in the `/mnt/ec2-fs/etc/sysconfig/network` file so that networking starts:

    ```
    NETWORKING=yes
    ```

3. Add the following lines to `/mnt/ec2-fs/etc/fstab` so that local disk storage on `/dev/sda2` and swap space on `/dev/sda3` are mounted at system startup:

```
/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 Amazon EC2 Instance Storage (p. 479).

4. Allocate appropriate system run levels so that all your required services start at system startup. For example, to enable the service `my-service` on multiuser and networked run levels, enter the following commands:

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

Your new installation is successfully installed and configured to operate in the Amazon EC2 environment.

5. Enter the following commands to umount the image:

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

The `<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
```

Bundle the Loopback File Image

To bundle the loopback file image

• Enter the following command:

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

The `<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 ID associated with your AWS account.

Note

The user ID is your AWS account ID without dashes. It consists of 12 to 15 characters, and it's *not* the same as your Access Key ID. For information about viewing your account ID, see Viewing Your Account ID (p. 353).

Example

This command bundles an image created in a loopback file.

```
# ec2-bundle-image -k pk-HKZYKTAIG2ECMXYIBH3HXV4ZBEXAMPLE.pem -c cert-HKZYK
TAIG2ECMXYIBH3HXV4ZBEXAMPLE.pem -u 111122223333 -i image.img -d bundled/ -p
fred -r x86_64
 Please specify a value for arch [i386]:
 Bundling image file...
 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...
 ec2-bundle-image complete.
```

Upload a Bundled AMI

You must upload the bundled AMI to Amazon S3 before Amazon EC2 can access it. This task is necessary when you create Amazon EC2 instance store-backed AMIs from an existing file or from a loopback. 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 AWS account, the `ec2-upload-bundle` command will fail.

To upload the bundled AMI

*   Enter the following command:

```
# ec2-upload-bundle -b <bucket> -m <manifest_path> -a <access_key> -s
<secret_key>
```

The `<bucket>` is the target bucket (you can bundle to a "subfolder"; e.g., myawsbucket/ami-image-folder), `<access_key>` is your AWS Access Key, and `<secret_key>` is your AWS Secret Key. The `-m <manifest_path>` is the full path to the manifest file (e.g., /tmp/image.manifest.xml).

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.

Register the AMI

Your must register your image with Amazon EC2, so Amazon EC2 can locate it and run instances based on it. This task is necessary when you create Amazon EC2 instance store-backed AMIs from an existing file or from a loopback.

Note

If you make any changes to the source image stored in Amazon S3, you must reregister the image.

To register the AMI that you created and uploaded to Amazon S3

*   Enter the following command (note that it is part of the Amazon EC2 API tools, and not the AMI tools):

```
PROMPT> ec2-register <your-s3-bucket>/image.manifest.xml -n image_name
IMAGE ami-2bb65342
```

Amazon EC2 returns an AMI identifier, the value next to the `IMAGE` tag (`ami-2bb65342` in the example), that you can use to run instances.

Creating Paid AMIs

Topics

A *paid AMI* is an AMI that you sell to other Amazon EC2 users. These users pay you according to the price you set. AWS Marketplace and Amazon DevPay provide capabilities to create and distribute paid AMIs. This section gives an introduction to the AWS Marketplace and Amazon Devpay.

The AWS Marketplace is an online store that helps customers find, compare, and immediately start using the software they need to build products and run their businesses. For an overview of the AWS Marketplace, go to https://aws.amazon.com/marketplace/help. You can also visit the AWS Marketplace.

Sellers interested in submitting AMIs to AWS Marketplace should consult the AWS Marketplace Seller's Guide.

AWS Marketplace uses the same approach to validate product codes, see Confirming an Instance Is Running with a Product Code (p. 59) and Getting the Product Code from Within an Instance (p. 60) . However, other details of creating, submitting AMIs to the AWS Marketplace are different, and are discussed in more detail in the AWS Marketplace Seller's Guide.

Important

Amazon DevPay does not support Amazon EBS-backed AMIs. All paid AMIs must be backed by Amazon instance store.

How Is AWS Marketplace different than DevPay?

There are substantial differences between AWS Marketplace and Amazon DevPay. Both help customers buy software that runs on AWS, but AWS Marketplace offers a more comprehensive experience. For software buyers the key differences are:

- AWS Marketplace offers a more Amazon.com-like shopping experience, simplifying discovery of available software.
- AWS Marketplace products work with other AWS features such as VPC and can be run on Reserved and Spot instances, in addition to normal On Demand Instances.
- AWS Marketplace supports EBS-backed software, where DevPay does not.

Amazon DevPay

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.

> **Important**
>
> Amazon DevPay does not support Amazon EBS-backed AMIs. All paid AMIs must be backed by Amazon instance store.

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

- You can charge customers for your Amazon EC2 instance store-backed 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 Amazon EC2 instance store-backed 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 Amazon EC2 instance store-backed 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 a fixed transaction fee and pays you the difference.
4	You pay the costs of Amazon EC2 that your Amazon EC2 instance store-backed AMI used, and a percentage-based DevPay fee.

For more information about Amazon DevPay, refer to the *Amazon DevPay Developer Guide*.

Summary of How Paid AMIs Work with Amazon DevPay

With a paid AMI using Amazon DevPay, your customers:

- Must be signed up to use Amazon EC2 themselves
- Buy your paid Amazon EC2 instance store-backed AMI and then launch instances of it
- Always use *their own* AWS credentials when launching instances; you don't launch instances of your paid AMI for them with your credentials
- Pay the price you set for the paid AMI, and not the normal Amazon EC2 rates

Important

The discounts you get with Amazon EC2 Reserved Instances don't apply to Amazon DevPay products. That is, if you purchase Reserved Instances, you don't get the lower price associated with them when your customers launch your paid or supported AMIs. Also, if your customers purchase Reserved Instances, when they use your paid or supported AMIs, they continue to pay the price you specified for the use of your paid or supported AMIs. For more information about Reserved Instances, see Reserved Instances (p. 195).

You can also use Amazon EC2 and Amazon DevPay together with a *supported AMI*. For more information about supported AMIs, see Supported AMIs (p. 60).

The following figure and table 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*.
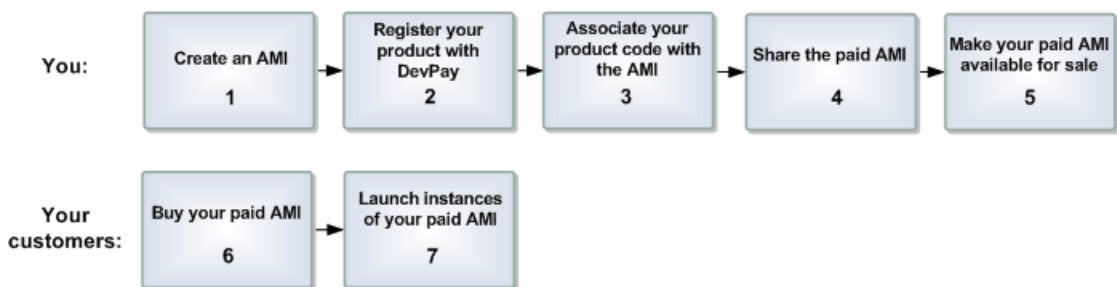
Paid AMI Process

1	You create an Amazon EC2 instance store-backed AMI as described elsewhere in this guide.

2	You register a product with Amazon DevPay.
	For more information, see Product Registration (p. 57). As part of this process, you provide a product description, product pricing, etc. This registration process creates a product code for the product and a URL where customers can sign up to use the product (called the *purchase URL*).
	Note
	You cannot register an AMI that already has an AWS Marketplace code associated with it.
3	You use an Amazon EC2 command or API call to associate the product code with your Amazon EC2 instance store-backed AMI.
	For more information, see Associating a Product Code with an AMI (p. 58). This makes the AMI a paid AMI.
4	You use an Amazon EC2 command or API call to share the Amazon EC2 instance store-backed AMI with select customers or the public.
	For more information, see Sharing Your Paid AMI (p. 59).
	Note
	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).
5	You make your paid AMI available for sale.
	To do this, you make the aforementioned purchase URL available. You can advertise your paid AMI in the Solutions Catalog on the AWS Developer Connection site and on the Amazon Machine Images (AMIs) page on the AWS Resource Center.
6	Customers 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 step 5).
7	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 Paid AMIs (p. 31).

Note

You can associate your DevPay product code with more than one Amazon EC2 instance store-backed AMI. However, a single AMI can be associated with only one product code. If you plan to sell multiple AMIs, you could sell them all under a single product code, or different product codes (by registering multiple DevPay products). For information about why you might choose a single product code or multiple product codes, go to If You Have Multiple AMIs to Sell in the *Amazon DevPay Developer Guide*.

Each customer's bill for the AMI is displayed on their Application Billing page, which shows the activity for DevPay products. Also, at any time, you can confirm the customer is still currently subscribed to your product. For more information, refer to the *Amazon DevPay Developer Guide*.

Note

In the preceding process, you associate your product code with your own Amazon EC2 instance store-backed AMI and sell the AMI as a DevPay product. There's another scenario for using

DevPay with Amazon EC2 in which you sell software or a service to EC2 users and let them associate your product code with their own Amazon EC2 instance store-backed AMIs. For more information, see Supported AMIs (p. 60).

The Product Code and AMI Rebundling

Associating a product code with an AMI turns it into a paid AMI that EC2 users must sign up for to use. Can you ensure that the product code stays with the AMI if someone rebundles the AMI? The answer varies for Linux/UNIX AMIs and Windows AMIs. These are described in the following sections.

Linux/UNIX AMIs

If you give the customer root access to your paid Linux/UNIX AMI, the customer can rebundle it (for more information, see Creating Your Own AMIs (p. 35)). If your customer uses AWS tools to rebundle the AMI, the rebundled AMI inherits the product code. When launching instances of the rebundled AMI, the customer is still billed for usage based on your price. However, if the customer doesn't use the AWS tools when rebundling, the rebundled AMI won't inherit the product code, and the customer will pay normal Amazon EC2 rates and not your price. Also, a customer with root access could find some other way to remove the product code from the AMI.

When a customer contacts you for support for a paid AMI, you can confirm your product code is associated with the AMI and the customer's instance is currently running the AMI. For more information, go to Confirming an Instance Is Running with a Product Code (p. 59).

If you have software installed on the AMI, the software can retrieve the instance metadata to determine if the product code is associated with the instance. For more information, see Getting the Product Code from Within an Instance (p. 60).

Keep in mind that the preceding methods for confirming the association of the product code with the instance are not foolproof because a customer with root access to the instance could return false information indicating the product code is associated with the instance.

Windows AMIs

When you associate a product code with a Windows AMI, the association is permanent. Therefore, we recommend you keep a separate, base copy of the AMI that has no product code associated with it.

Anyone who purchases a Windows AMI can rebundle it (for more information, see Creating Your Own AMIs (p. 35)). The product code is automatically transferred to the rebundled AMI. When EC2 users launch instances of the rebundled AMI, they pay the rates you set when you registered your DevPay product. In turn, you're charged for the EC2 costs they incur.

Product Registration

You must register a product with Amazon DevPay. The product can cover a single Amazon EC2 instance store-backed AMI that you want to sell or multiple Amazon EC2 instance store-backed AMIs. During registration, you provide product information such as pricing, and you receive information you need to sell your product.

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.

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)
- The specific Regions, environments, and instance types the product covers
- Pricing for use of the product (you can set different prices based on Region, environment, and instance type)

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

Registration provides you with the following information:

- Product code, which contains a product code and type
- 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 with Amazon DevPay (p. 55) and Supported AMIs (p. 60). 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, refer to the *Amazon DevPay Developer Guide*.

> **Note**
>
> AWS must approve your product after you register it. The approval process typically takes one business day.

Associating a Product Code with an AMI

You must be the owner of an Amazon EC2 instance store-backed AMI to associate a product code with it. Each AMI can have only a single product code associated with it, but you can associate a single product code with more than one 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>
```

The `<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-2bb65342 AMI with the 774F4FF8 product code.

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

productCodes         ami-2bb65342          productCode   [devpay: 774F4FF8]
```

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

```
PROMPT>    ec2-describe-image-attribute ami-2bb65342 --product-code

productCodes         ami-2bb65342          productCode   [devpay: 774F4FF8]
```

Sharing Your Paid AMI

After you associate the product code with the Amazon EC2 instance store-backed 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
```

The *<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-2bb65342 AMI with the public.

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

launchPermission         ami-2bb65342     ADD      group    all
```

Confirming an Instance Is Running 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. This applies to both DevPay AMIs and AWS Marektlace AMIs.

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 instance is running an AMI associated with your product code (DevPay or AWS Marketplace)

- Enter the following command:

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

The `<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 it has an Amazon DevPay or AWS Marketplace product code. The instance retrieves the product code similarly to how retrieves other metadata. For more information about retrieving metadata, see Instance Metadata (p. 302).

To retrieve a product code, query a web server with this REST-like API call:

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

Amazon EC2 returns a response similar to the following:

```
774F4FF8
```

Supported AMIs

Supported AMIs are different from paid AMIs. With a supported AMI, you charge for software or a service you provide that customers use with their own AMIs.

The main difference between a paid AMI and a *supported AMI* is how the AMI is associated with a product code:

- **Paid AMI—**You associate your own product code with your own AMI
- **Supported AMI—**Other EC2 users associate your product code with their own AMIs

Important

If your customers purchase Reserved Instances, they don't get the Reserved Instance price discount with supported AMIs. That is, if they associate your product code with their AMIs, they don't get the lower price associated with their Reserved Instances when they launch those AMIs.

They always pay the price that you specified for your DevPay product. For more information, see Reserved Instances (p. 195).

The following figure and table summarizes the flow for creating and using supported AMIs.

Supported AMI Process

1	You register a product with Amazon DevPay.
	For more information, see Product Registration (p. 57). As part of this process, you provide a product description, product pricing, etc. This registration process creates a product code for the product and a URL where customers can sign up to use the product (called the *purchase URL*).
2	You make your product available for sale.
3	Customers 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 step 2).
4	Customers then use an Amazon EC2 command or API call to associate the product code with their Amazon EC2 instance store-backed AMIs.
	For more information, see Associating a Product Code with an AMI (p. 58).
5	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.

Note

Amazon EC2 prevents your customers (but not you as the product code owner) from associating your product code with AMI types the product isn't configured for. For example, if the product is configured only for Linux/UNIX AMIs, your customers can't associate the product code with Windows AMIs. Also, Amazon EC2 prevents your customers from launching specific instance types your product isn't configured for. For more information about product configuration, go to Your Product's Configuration and Price in the *Amazon DevPay Developer Guide*.

Each customer's bill for the AMI is displayed on their Application Billing page, which shows the activity for DevPay products. For more information, refer to 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. For more information, see Confirming an Instance Is Running with a Product Code (p. 59).

Sharing AMIs Safely

Topics

- Protecting a Shared AMI (Linux/UNIX) (p. 62)
- Sharing AMIs (p. 66)

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

For additional information about sharing AMIs safely, go to the following articles on the AWS Developer Resources website.

- How To Share and Use Public AMIs in A Secure Manner
- Public AMI Publishing: Hardening and Clean-up Requirements

For information on building shared AMIs, see Protecting a Shared AMI (Linux/UNIX) (p. 62). For information on sharing AMIs, see Sharing AMIs (p. 66).

Protecting a Shared AMI (Linux/UNIX)

Following these guidelines produces a better user experience, makes your users' *instances* less vulnerable to security issues, and helps protect you.

To build a shared AMI, follow these guidelines:

Shared AMI Guidelines

1	Update the AMI Tools at Boot Time (p. 62)
2	Disable Password-Based Logins for Root (p. 63)
3	Install Public Key Credentials (p. 64)
4	Disabling sshd DNS Checks (optional) (p. 65)
5	Identify Yourself (p. 65)
6	Protect Yourself (p. 65)
7	Protect Paid AMIs (p. 66)

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

For AMIs backed by instance store, 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 preceding 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 locate the following line:

```
#PermitRootLogin yes
```

2. Change the line 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.

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

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

Note

This step assumes that a /root/firstboot file is bundled with the image. If the 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.

Remove SSH Host Key Pairs

If you plan to share an AMI derived from a public AMI, remove the existing SSH host key pairs located in `/etc/ssh`. This forces SSH to generate new unique SSH key pairs when someone launches an instance using your AMI, improving security and reducing the likelihood of "man-in-the-middle" attacks.

The following list shows the SSH files to remove.

- ssh_host_dsa_key
- ssh_host_dsa_key.pub
- ssh_host_key
- ssh_host_key.pub
- ssh_host_rsa_key
- ssh_host_rsa_key.pub

Important

If you forget to remove the existing SSH host key pairs from your public AMI, our routine auditing process will notify you and all customers running instances of your AMI of the potential security risk. After a short grace period, we will mark the AMI private.

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` or `ImportKeyPair`) 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/latest//meta-data/public-keys/0/openssh-key >
/tmp/my-key
if [ $? -eq 0 ] ; then
        cat /tmp/my-key >> /root/.ssh/authorized_keys
        chmod 700 /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.

Disabling 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

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

```
#UseDNS yes
```

2. Change the line 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 because each AMI is represented by an account 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. You can also post the AMI to the Amazon Machine Images (AMIs) page.

Protect Yourself

The previous sections described how to make your shared AMIs safe, secure, and usable 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 instance 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.

Protect Paid AMIs

The simplest way to prevent users from rebundling paid AMIs that you create is to not provide root access to the AMI and to pay attention to security announcements that involve privilege escalations. Amazon EC2 requires you to have root access on any AMI that you rebundle.

If you must provide root access to an AMI, Amazon EC2 tools are designed to protect the product code. Although this is effective, it is not guaranteed and users might create AMIs using other tools.

To ensure users cannot rebundle your paid AMIs, we recommend that you configure your application to check the instance metadata to verify that the product code is intact.

Sharing AMIs

Amazon EC2 enables you to share your AMIs with other AWS accounts. 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 Protecting a Shared AMI (Linux/UNIX) (p. 62) section.

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

The `launchPermission` attribute is a list of accounts and *launch groups*. *Launch permissions* can be granted by adding or removing items from the list. Explicit launch permissions for accounts 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 AWS accounts. The accounts launching the AMI are billed.

Making an AMI Public

This section explains how to make an AMI public.

> **Note**
>
> Images with an AWS Marketplace product code cannot be made public.

To make an AMI public

- Add the `all` group to the AMI's `launchPermission`.

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

The `<ami_id>` parameter is the ID of the AMI.

To check the launch permissions of an AMI

- Enter the following command, where `<ami_id>` is the ID of the AMI.

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

To make an AMI private again

- Remove the `all` group from its launch permissions, where `<ami_id>` is the ID of the AMI.

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

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 AWS Accounts

You can share an AMI with specific AWS accounts without making the AMI public. All you need is the account ID.

To grant explicit launch permissions

- Enter the following command:

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

The *<ami_id>* is the ID of the AMI and *<user_id>* is the account ID, without hyphens.

To remove launch permissions for an account

- Enter the following command:

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

The *<ami_id>* is the ID of the AMI and *<user_id>* is the 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
```

The *<ami_id>* is the ID of the AMI.

> **Note**
>
> The AMI owner always has rights to the AMI and is unaffected by this command.

Example

The following example grants launch permissions to the AWS account with ID 111122223333 for the ami-2bb65342 AMI:

```
PROMPT> ec2-modify-image-attribute ami-2bb65342 -l -a 111122223333
launchPermission        ami-2bb65342    ADD     userId  111122223333
```

The following example removes launch permissions from the AWS account with ID 111122223333 for the ami-2bb65342 AMI:

```
PROMPT> ec2-modify-image-attribute ami-2bb65342 -l -r 111122223333
launchPermission        ami-2bb65342    REMOVE  userId  111122223333
```

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 you create a shared AMI, you can publish information about it in the Amazon EC2 Resource Center.

To publish your AMI

1. Post it in the Public AMIs Folder of the Amazon Web Services Resource Center, and include the following information:

   - AMI ID
   - AMI name (for Amazon EBS-backed AMIs) or AMI manifest (for Amazon EC2 instance store-backed AMIs)
   - Publisher
   - Publisher URL
   - OS / Distribution
   - Key Features
   - Description
   - Daemons / Services
   - Release Notes

2. If you want to, you can paste the following information into the document. You must be in HTML edit mode.

```
<strong>AMI ID: </strong>[ami-id]<br />
<strong>AMI Manifest: </strong>[myawsbucket/image.manifest.xml]<br />
<h2>About this &AMI;</h2>
<ul>

    <li>Published by [Publisher] (<a href="http://www.example.com">[ht
tp://www.example.com]</a>).<br />
            </li>
    <li>[Key Features] <br />
            </li>
    <li>[Description]</li>
    <li>This image contains the following daemons / services:
    <ul>

        <li>[Daemon 1]</li>
        <li>[Daemon 2]</li>
    </ul>
            </li>
</ul>
<h2><strong>What&#39;s New?</strong></h2>The following changes were made on
 [Date].<br />
<ul>

    <li>[Release Notes 1]</li>
</ul>
<span style="font-size: x-small; font-family: courier new,couri
er">     - [Note 1]</span><br />
<span style="font-size: x-small; font-family: courier new,couri
er">     - [Note 2]</span><br />
<span style="font-size: x-small; font-family: courier new,couri
er">     - [Note 3]</span><br />
<ul>
```

Amazon Linux AMIs

The Amazon Linux AMI is a supported and maintained Linux image provided by Amazon Web Services (AWS) for use on Amazon Elastic Compute Cloud (Amazon EC2). It is designed to provide a stable, secure, and high-performance execution environment for applications running on Amazon EC2. It also includes packages that enable easy integration with AWS, including launch configuration tools and many popular AWS libraries and tools. Amazon Web Services provides ongoing security and maintenance updates to all instances running the Amazon Linux AMI. The Amazon Linux AMI is provided at no additional charge to Amazon EC2 users.

Topics

Finding the Amazon Linux AMI

For a list of the latest Amazon Linux AMIs, go to Amazon Linux AMI.

Launching and Connecting to an Amazon Linux Instance

After locating your desired AMI, note the AMI ID. You can use the AMI ID to launch and then connect to your instance.

The Amazon Linux AMI does not allow remote root SSH by default. Also, password authentication is disabled to prevent brute-force password attacks. To enable SSH logins to a running Amazon Linux AMI, you must provide your key pair to the instance at launch. You must also set the security group used to launch your instance to allow SSH access. By default, the only account that can log in remotely via SSH is ec2-user. The ec2-user has sudo privileges. If you want to enable remote root log in, please be aware that it is less secure than relying on key pairs and a secondary user.

For information on launching and using your Amazon Linux instance, go to Launching and Using Instances. For information on connecting to your Amazon Linux instance, go to Connecting to Linux/UNIX Instances from Linux/UNIX.

Identifying Amazon Linux AMI Images

Each image contains a unique `/etc/image-id` that identifies the AMI. This file contains information about the image.

Following is an example of the `/etc/image-id` file:

```
# cat /etc/image-id
image_name="amzn-ami-pv"
image_version="2012.03"
image_arch="x86_64"
image_file="amzn-ami-pv-2012.03.1.x86_64.ext4"
image_stamp="b0e1-f3cd"
image_date="20120324170052"
recipe_name="amzn ami"
recipe_id="2aaca5dd-b34f-4a46-8281-d2471ce38631"
```

The `image_name`, `image_version`, and `image_arch` items come from the build recipe that Amazon used to construct the image. The `image_stamp` is simply a unique random hex value generated during image creation. The `image_date` item is in YYYYMMDDhhmmss format, and is the UTC time of image creation. The `recipe_name` and `recipe_id` refer to the name and ID of the build recipe Amazon used to construct the image, which identifies the current running version of the Amazon Linux AMI. This file will not change as you install updates from the yum repository.

Amazon linux AMIs contain a `/etc/system-release` file that specifies the current release that is installed. This file is updated through yum and is part of the system-release `rpm`.

Following is an example of a `/etc/system-release` file:

```
# cat /etc/system-release
Amazon Linux AMI release 2012.03
```

An Amazon Linux AMI also contains a machine readable version of the `/etc/system-release` file found in `/etc/system-release-cpe` and follows the CPE specification from MITRE (CPE).

Included AWS Command Line Tools

The following popular command line tools for AWS integration and usage have been included in the Amazon Linux AMI or in the repositories:

- aws-amitools-ec2
- aws-apitools-as
- aws-apitools-cfn
- aws-apitools-common
- aws-apitools-ec2
- aws-apitools-elb
- aws-apitools-iam
- aws-apitools-mon
- aws-apitools-rds
- aws-cfn-bootstrap

- aws-scripts-ses

To simplify the configuration of these tools, a simple script has been included to prepare
AWS_CREDENTIAL_FILE, JAVA_HOME, AWS_PATH, PATH, and product-specific environment variables
after a credential file has been installed.

Also, to allow the installation of multiple versions of the API and AMI tools, we have placed symlinks to
the desired versions of these tools in `/opt/aws`, as described here:

- /opt/aws/bin—Symlink farm to /bin directories in each of the installed tools directories.

- /opt/aws/{apitools|amitools}—Products are installed in directories of the form [name]-version and symlink
  [name] attached to the most recently installed version.

- /opt/aws/{apitools|amitools}/[name]/environment.sh—Used by /etc/profile.d/aws-apitools-common.sh
  to set product-specific environment variables (EC2_HOME, etc.).

Cloud-init

Cloud-init is an open source application built by Canonical that is used to bootstrap Linux images in a
cloud computing environment such as Amazon EC2. The Amazon Linux AMI contains a customized
version of Cloud-init. It enables you to specify actions that should happen to your instance at boot time.
You can pass desired actions to Cloud-init through the user data fields when launching an instance. This
means you can use common AMIs for many use cases and configure them dynamically at startup. The
Amazon Linux AMI also uses Cloud-init to perform initial configuration of the ec2-user account.

For more information about Cloud-init, go to https://help.ubuntu.com/community/CloudInit.

The Amazon Linux AMIs use the following Cloud-init actions (configurable in /etc/sysconfig/cloudinit):

- action: INIT (always runs)
  - Setting a default locale.
  - Setting the hostname.
  - Parsing and handling user data.
- action: CONFIG_SSH
  - Generating host private SSHkeys.
  - Adding user's public SSHkeys to .ssh/authorized_keys for easy login and administration.
- action: PACKAGE_SETUP
  - Preparing yum repo.
  - Handles package actions defined in user data.
- action: RUNCMD
  - Runs a shell command.
- action: RUN_USER_SCRIPTS
  - Executes user scripts found in user data.
- action: CONFIG_MOUNTS
  - Mounts ephemeral drives.
- action: CONFIG_LOCALE
  - Sets the locale in the locale config file according to user data.

Supported User-Data Formats

Cloud-init supports user-data handling of a variety of formats:

- Gzip
  - If user-data is gzip compressed, Cloud-init will decompress the data and handle as appropriate.
- MIME multipart
  - Using a MIMI multipart file you can specify more than one type of data. For example, you could specify both a user-data script and a cloud-config type. Each part of the multipart file can be handled by Cloud-init if it is one of the supported formats.
- Base64 decoding
  - If user-data is base64 encoded, Cloud-init determines if it can understand the decoded data as one of the supported types. If it understands the decoded data, it will decode the data and handle as appropriate. If not, it returns the base64 data intact.
- User-Data script
  - Begins with "#!" or "Content-Type: text/x-shellscript".
  - The script will be executed by "/etc/init.d/cloud-init-user-scripts" level during first boot. This occurs late in the boot process (after the initial configuration actions were performed).
- Include file
  - Begins with "#include" or "Content-Type: text/x-include-url".
  - This content is an include file. The file contains a list of URLs, one per line. Each of the URLs will be read, and their content will be passed through this same set of rules. The content read from the URL can be gzipped, MIME-multi-part, or plain text.
- Cloud Config Data
  - Begins with "#cloud-config" or "Content-Type: text/cloud-config".
  - This content is cloud-config data. See the examples for a commented example of supported config formats.
- Cloud Boothook
  - Begins with "#cloud-boothook" or "Content-Type: text/cloud-boothook".
  - This content is boothook data. It is stored in a file under /var/lib/cloud and then executed immediately.
  - This is the earliest "hook" available. Note that there is no mechanism provided for running it only once. The boothook must take care of this itself. It is provided with the instance ID in the environment variable INSTANCE_ID. Use this variable to provide a once-per-instance set of boothook data.

Repository Configuration

Beginning with the 2011.09 release of the Amazon Linux AMI, the repository structure is configured such that, by default, you receive a continuous flow of updates that take you from one version of the Amazon Linux AMI to the next. For example, with the release of the 2012.03 Amazon Linux AMI, if you launch a 2011.09 AMI and run `yum update -y`, it'll be the same as launching the 2012.03 AMI and running `yum update -y`.

Non-Amazon Linux operating systems segregate their repositories from one version to the next. Whereas Amazon Linux AMIs are treated as snapshots in time, with a repository and update structure that, by default, gives you the latest packages.

With Amazon Linux AMI you can choose to not to accept the latest packages by enabling the *lock-on-launch* feature. The lock-on-launch feature locks your newly launched instance to receive updates from the specified AMI package. For example, you can launch a 2011.09 AMI and have it receive only the updates

that were released prior to the 2012.03 Amazon Linux AMI, until you are ready to migrate to the 2012.03 AMI. The lock-on-launch feature takes advantage of cloud-init to implement this use case. To enable this feature in new instances, launch a 2011.09 Amazon Linux AMI with the following user data passed to cloud-init, either via the EC2 console or via `ec2-run-instances` command with the the the `-f` flag:

```
#cloud-config
repo_releasever:2011.09
```

To lock existing 2011.09 instances to the 2011.09 repositories

1. edit `/etc/yum.conf`.
2. comment out `releasever=latest`.
3. run `yum clean all` to clear the cache.

Adding Packages

The Amazon Linux AMI is designed to be used with online package repositories hosted in each Amazon EC2 region. These repositories provide ongoing updates to packages in the Amazon Linux AMI, as well as access to hundreds of additional common open source server applications. The repositories are available in all regions and are accessed via yum update tools. Hosting repositories in each region allows updates to be deployed quickly and without any data transfer charges. The packages can be installed by issuing yum commands such as the following example:

```
#   sudo yum install httpd
```

Access to the Extra Packages for Enterprise Linux (EPEL) repository is configured, but it is not enabled by default. EPEL provides third-party packages in addition to those that are in the Amazon Linux AMI repositories. The third-party packages are not supported by Amazon Web Services.

> **Important**
>
> Instances running in an Amazon Virtual Private Cloud (VPC) need to have an Internet Gateway attached to the VPC to contact the yum repository. For more information on Amazon VPC, go to the Amazon Virtual Private Cloud User Guide.

If you find that the Amazon Linux AMI does not contain an application you need, you can simply install the application directly on your Amazon Linux AMI instance. The Amazon Linux AMI uses RPM and yum for package management, and that will likely be the simplest way to install new applications. You should always check to see if an application is available in our central Amazon Linux AMI repository first because many applications are available there. These applications can easily be added to your AMI instance.

To upload your applications onto an Amazon Linux running instance, use `scp` or `sftp` and then configure the application by logging on to your instance. Your applications can also be uploaded during the instance launch by using the `PACKAGE_SETUP` action from built-in Cloud-init package. For more information, see Cloud-init.

Accessing Source Packages for Reference

You can view the source of packages you have installed inside Amazon EC2 for reference purposes by using tools provided in the Amazon Linux AMI. Source packages are available for all of the packages included in the Amazon Linux AMI and the online package repository. Simply determine the package name for the source package you want to install and use the get_reference_source command to view source within your running instance. For example:

```
# get_reference_source -p  httpd
```

Following is a sample response:

```
Requested package: httpd
Found package from local RPM database: httpd-2.2.22-1.23.amzn1.x86_64
Corresponding source RPM to found package : httpd-2.2.22-1.23.amzn1.src.rpm

Are these parameters correct? Please type 'yes' to continue: yes
Source RPM downloaded to: /usr/src/srpm/debug/httpd-2.2.22-1.23.amzn1.src.rpm
```

The source RPM will be placed in the /usr/src/srpm/debug directory of your running Amazon EC2 instance. From there it can be unpacked, and, for reference, you can view the source tree using standard RPM tools. After you finish debugging, the package will be available for use in Amazon EC2.

Important

Instances running in an Amazon Virtual Private Cloud (Amazon VPC) need to have an Internet Gateway attached to the VPC to contact the yum repository. For information on Amazon VPC, go to the Amazon Virtual Private Cloud User Guide.

Developing Applications

A full set of Linux development tools is provided in the yum repository for the Amazon Linux AMI. To develop applications on the Amazon Linux AMI, simply select the development tools you need with yum. Alternatively, many applications developed on CentOS and other similar distributions should run on the Amazon Linux AMI.

Instance Store Access

The instance store drive ephemeral0 is mounted in /media/ephmeral0 only on Amazon instance store-backed AMIs. This is different than many other images that mount the instance store drive under /mnt.

Product Life Cycle

The Amazon Linux AMI is updated regularly with security and feature enhancements. If you do not need to preserve data or customizations on your running Amazon Linux AMI instances, you can simply relaunch new instances with the latest updated Amazon Linux AMI. If you do need to preserve data or customizations on your running Amazon Linux AMI instances, you can maintain those instances through the Amazon Linux AMI yum repositories. The yum repositories contain all the updated packages. You can chose to apply these updates to your running instances.

Older versions of the AMI and update packages will continue to be available for launch in Amazon EC2, even as new Amazon Linux AMI versions are released. However, in some cases, if you're seeking support for an older version of the Amazon Linux AMI through Amazon Premium Support, we might ask you to move to newer versions as part of the support process.

Important

Instances running in an Amazon Virtual Private Cloud (Amazon VPC) need to have an Internet Gateway attached to the VPC to contact the yum repository. For information on adding an Internet Gateway to your Amazon VPC, go to the Amazon Virtual Private Cloud User Guide.

Security Updates

Security updates are provided via the Amazon Linux AMI yum repositories as well as via updated Amazon Linux AMIs. Security alerts will be published in the Amazon Linux AMI Security Center. For more information on AWS security policies or to report a security problem, visit the AWS Security Center.

Amazon Linux AMIs are configured to download and install security updates at launch time. This is controlled via a Cloud-init setting called `repo_upgrade`. The following snippet of cloud-init configuration shows how you can change the settings in the user data text you pass to your instance initialization:

```
#cloud-config
repo_upgrade:security
```

The possible values for the `repo_upgrade` setting are as follows:

- security
  - Apply outstanding updates that Amazon marks as security updates.
- bugfix
  - Apply updates that Amazon marks as bug fixes. Bug fixes are a larger set of updates, which include security updates and fixes for various other minor bugs.
- all
  - Apply all applicable available updates, regardless of their classification.
- none
  - Do not apply any updates to the instance on startup.

The default setting for `repo_upgrade` is security. That is, if you don't specify a different value in your user data, by default the Amazon Linux AMI will perform the security upgrades at launch for any packages installed at that time. Amazon Linux AMI will also notify you of any updates to the installed packages by listing the number of available updates upon login using the `motd`. To install these updates, you will need to run `sudo yum upgrade` on the instance.

Important

Instances running in an Amazon Virtual Private Cloud (Amazon VPC) need to have an Internet Gateway attached to the VPC to contact the yum repository. For information on adding an Internet Gateway to Amazon VPC, go to the Amazon Virtual Private Cloud User Guide.

Support

Support for installation and use of the base Amazon Linux AMI is included through subscriptions to AWS Premium Support. For more information, go to Premium Support.

You're encouraged to post any questions you have on using the Amazon Linux AMI to the Amazon EC2 forums.

You can report bugs either to Premium Support or the Amazon EC2 forums.

Enabling Your Own Linux Kernels

Topics

Amazon EC2 allows you to load a paravirtual Linux kernel within an Amazon Machine Image (AMI) or Amazon EBS volume. You have the option to create images that contain a kernel and initrd (initial RAM disk), and behave in a manner that is closer to traditional virtual or physical Linux installations. By enabling you to boot from the kernel within volumes, this feature allows you to seamlessly upgrade the kernel on Amazon EC2 instances. We expect that AMI providers will update their AMIs to use this new feature, and most Amazon EC2 users will be able to begin managing their own kernels when these updated AMIs become available. Your AMI provider can tell you when it plans to support this feature. However, if you want to begin managing your own kernel now, the following section shows how. This process assumes general knowledge of Amazon EC2 AMI bundling and registration, as well as knowledge of how to install kernel packages and configure GRUB on your Linux systems.

PV-GRUB: A New Amazon Kernel Image

To enable user-provided kernels, Amazon has published Amazon Kernel Images (AKIs) that use a system called PV-GRUB. PV-GRUB is a paravirtual "mini-OS" that runs a version of GNU GRUB, the standard Linux boot loader. PV-GRUB selects the kernel to boot by reading /boot/grub/menu.lst from your image. It will load the kernel specified by your image and then shut down the "mini-OS," so that it no longer consumes any resources. One of the advantages of this solution is that PV-GRUB understands standard grub.conf or menu.lst commands, which allows it to work with most existing Linux distributions.

The following task list describes what you need to do to enable an AMI to use PV-GRUB AKI to run a user-provided kernel.

Enabling an AMI to Run A User-Provided Kernel

1	Install an Amazon EC2-compatible kernel.
2	Generate an initrd.
3	Populate /boot/grub/menu.lst referencing your kernel.
4	Select an appropriate AKI ID from the Amazon Kernel Image IDs section that follows.
5	Bundle the AMI and set the default to your chosen AKI.
6	Upload and register your new AMI.
7	For existing AMIs, you can simply specify the appropriate AKI ID when you call RunInstances or when you use the AWS Management Console.

Note

To update an existing AMI to use a user-provided kernel, re-launch the AMI and follow steps 1 through 6 specified in the preceding task list.

For procedures associated with the preceding task list, see Using the User-Provided Kernel (p. 81).

Configuring the GRUB

In order for PV-GRUB to boot, a GRUB menu.lst file must exist in the image. For most distributions, you have two options for the GRUB configuration:

- **Option 1:** Install GRUB and allow the default kernel installation scripts to handle the installing and updating the GRUB configuration. The steps necessary to install GRUB will vary depending on your Linux distribution, but typically GRUB will be available as a package you can install online.

- **Option 2:** Populate a general /boot/grub/menu.lst. An example of a menu.lst configuration file for booting an AMI with a PV-GRUB AKI follows.

   Important

   You must modify your own menu.lst for your specific environment.

```
default 0
timeout 3
fallback 1

title Vanilla EC2 Kernel 2.6.32.10
 root (hd0)
 kernel /boot/vmlinux-2.6.32.10-ACME_SYS_EC2 root=/dev/sda1
 initrd /boot/initrd-2.6.32.10-ACME_SYS_EC2

title Ubuntu EC2 2.6.32.302-EC
 root (hd0)
 kernel /boot/ubuntu-ec2 root=/dev/sda1
 initrd /boot/initrd-ec2
```

We recommend that you use option two to control the kernel booting for two reasons. First, Amazon EC2 users don't have interactive control over the boot process because there is no keyboard access. GRUB will proceed without user interaction. Second, and most important for Amazon EC2 instances, you want to protect against distributions that auto-update the default kernel and break your image. By not relying on the auto-update mechanism and explicitly choosing which kernel you run, you reduce the risk of an incompatible kernel becoming the default kernel.

A fallback kernel does not have to be specified in your menu.lst, but we recommend that you have a fallback when you test new new kernels. GRUB can fall back to another kernel in the event that the new kernel fails. Having a fallback kernel allows the instance to boot even if the new kernel is not found.

Amazon EBS Volumes on a PV-GRUB-Enabled AMI

There are two special things you should consider when you use a PV-GRUB-enabled image to mount EBS volumes. First, for Amazon EBS volumes the first partition must be a boot partition. Second, if you plan to use a logical volume manager (LVM) with Amazon EBS volumes, you need a separate boot partition outside of the LVM. Then you can create logical volumes with the LVM. PV-GRUB expects to find the menu.lst in /boot/grub. As a result, if the boot partition is mounted in at /boot, menu.lst will be found in /boot/boot/grub.

Amazon Kernel Image IDs

Several PV-GRUB AKIs are available depending on the type and location of your instance. There are AKIs for 32-bit and 64-bit architecture types, with each having one AKI for partitioned images and another AKI for partitionless images. You must choose an AKI with "hd0" in the name if you want a raw or

unpartitioned disk image (most images). Choose an AKI with "hd00" in the name if you want an image that has a partition table.

Most vendors, such as Fedora, Red Hat, Ubuntu, and Novell, use unpartitioned disk images. This means that they use the hd0 variants of PV-GRUB; almost without exception most users will want to use the hd0 variants.

Note

You cannot use the 64-bit version of PV-GRUB to start a 32-bit kernel or vice versa.

Note

You must not specify an Amazon ramdisk image (ARI) when using a PV-GRUB AKI.

The following AKI IDs should be used by users who are either registering new AMIs or who want to launch existing AMIs using PV-GRUB. Each AKI type is available in all Amazon EC2 Regions:

- **US-East-1**

  aki-88aa75e1 ec2-public-images/pv-grub-hd0_1.03-x86_64.gz.manifest.xml
  aki-b6aa75df ec2-public-images/pv-grub-hd0_1.03-i386.gz.manifest.xml
  aki-b4aa75dd ec2-public-images/pv-grub-hd00_1.03-x86_64.gz.manifest.xml
  aki-b2aa75db ec2-public-images/pv-grub-hd00_1.03-i386.gz.manifest.xml

- **US-West-1**

  aki-f77e26b2 ec2-public-images-us-west-1/pv-grub-hd0_1.03-x86_64.gz.manifest.xml
  aki-f57e26b0 ec2-public-images-us-west-1/pv-grub-hd0_1.03-i386.gz.manifest.xml
  aki-eb7e26ae ec2-public-images-us-west-1/pv-grub-hd00_1.03-x86_64.gz.manifest.xml
  aki-e97e26ac ec2-public-images-us-west-1/pv-grub-hd00_1.03-i386.gz.manifest.xml

- **US-West-2**

  aki-fc37bacc ec2-public-images-us-west-2/pv-grub-hd0_1.03-x86_64.gz.manifest.xml
  aki-fa37baca ec2-public-images-us-west-2/pv-grub-hd0_1.03-i386.gz.manifest.xml
  aki-f837bac8 ec2-public-images-us-west-2/pv-grub-hd00_1.03-x86_64.gz.manifest.xml
  aki-f637bac6 ec2-public-images-us-west-2/pv-grub-hd00_1.03-i386.gz.manifest.xml

- **EU-West-1**

  aki-71665e05 ec2-public-images-eu/pv-grub-hd0_1.03-x86_64.gz.manifest.xml
  aki-75665e01 ec2-public-images-eu/pv-grub-hd0_1.03-i386.gz.manifest.xml
  aki-8b655dff ec2-public-images-eu/pv-grub-hd00_1.03-x86_64.gz.manifest.xml
  aki-89655dfd ec2-public-images-eu/pv-grub-hd00_1.03-i386.gz.manifest.xml

- **AP-SouthEast-1**

  aki-fe1354ac ec2-public-images-ap-southeast-1/pv-grub-hd0_1.03-x86_64.gz.manifest.xml
  aki-f81354aa ec2-public-images-ap-southeast-1/pv-grub-hd0_1.03-i386.gz.manifest.xml
  aki-fa1354a8 ec2-public-images-ap-southeast-1/pv-grub-hd00_1.03-x86_64.gz.manifest.xml
  aki-f41354a6 ec2-public-images-ap-southeast-1/pv-grub-hd00_1.03-i386.gz.manifest.xml

- **AP-NorthEast-1**

  aki-44992845 ec2-public-images-ap-northeast-1/pv-grub-hd0_1.03-x86_64.gz.manifest.xml
  aki-42992843 ec2-public-images-ap-northeast-1/pv-grub-hd0_1.03-i386.gz.manifest.xml
  aki-40992841 ec2-public-images-ap-northeast-1/pv-grub-hd00_1.03-x86_64.gz.manifest.xml
  aki-3e99283f ec2-public-images-ap-northeast-1/pv-grub-hd00_1.03-i386.gz.manifest.xml

- **SA-East-1**

  aki-c48f51d9 ec2-public-images-sa-east-1/pv-grub-hd0_1.03-x86_64.gz.manifest.xml

aki-ca8f51d7 ec2-public-images-sa-east-1/pv-grub-hd0_1.03-i386.gz.manifest.xml
aki-c88f51d5 ec2-public-images-sa-east-1/pv-grub-hd00_1.03-x86_64.gz.manifest.xml
aki-ce8f51d3 ec2-public-images-sa-east-1/pv-grub-hd00_1.03-i386.gz.manifest.xml

- **US-Gov-West-1**

aki-79a4c05a ec2-public-images-gov-west-1/pv-grub-hd0_1.03-x86_64.gz.manifest.xml
aki-7ba4c058 ec2-public-images-gov-west-1/pv-grub-hd0_1.03-i386.gz.manifest.xml
aki-75a4c056 ec2-public-images-gov-west-1/pv-grub-hd00_1.03-x86_64.gz.manifest.xml
aki-77a4c054 ec2-public-images-gov-west-1/pv-grub-hd00_1.03-i386.gz.manifest.xml

Compatible PV-GRUB Kernels

There are a number of Linux distributions that have compatible Amazon EC2 kernels. The following is a brief, non-comprehensive list of kernels that we have worked with the maintainers to test:

- Fedora 8-9 Xen kernels
- Fedora 13 (2.6.33.6-147 and higher)
- Fedora 14 and later
- SLES/openSUSE 10.x, 11.0, 11.1 Xen
- SLES/openSUSE 11.x EC2 Variant
- Ubuntu EC2 Variant kernels
- Ubuntu 11.04 and later
- RHEL 5.x kernels
- RHEL 6.x kernels
- CentOS 5.x kernels
- CentOS 6.x kernels
- Gentoo (see FAQ)

It is possible that your specific Linux kernel will not boot using the new PV-GRUB method. If you find that to be the case, you should select a different kernel or use a non-PV-GRUB AKI to boot your instance.

> **Note**
>
> Kernels that disable the pv-ops XSAVE hypercall are known to work on all instance types, whereas those that enable this hypercall will fail to launch in some cases. Similarly, non-pv-ops kernels that do not adhere to the Xen 3.0.2 interface might fail to launch in some cases.

PV-GRUB Supported File Systems

Since PV-GRUB is a paravirtual version of GRUB 0.97, it has all the limitations of GRUB. Most importantly, this means that it will not work properly for certain disk layouts or file system types. The following are the /boot file systems that PV-GRUB can boot from:

- EXT2/3/4
- XFS
- ReiserFS
- BTRFS (beta)

Note

These are the /boot file systems that we have tested and verified. Others could boot from PV-GRUB, but haven't been tested.

Using the User-Provided Kernel

The following procedure gives an example of how to enable a openSUSE AMI to use the PV-GRUB AKI to run a user-provided kernel by rebundling from a running instance.

Important

The specific details of configuring your AMI to use PV-GRUB will vary depending on your exact Linux environment. The following example is for openSUSE 11.2.

To use the PV-GRUB AKI with an openSUSE AMI

1. Install an Amazon EC2-compatible kernel from the command line on your running Linux instance.

```
 # rpm –ivh /tmp/kernel-ec2-2.6.35-rc4.8.1.x86_64.rpm

 warning: /tmp/kernel-ec2-2.6.35-rc4.8.1.x86_64.rpm: Header V3 DSA signature:
 NOKEY, key ID a29f6635
Preparing...                        ###########################################
[100%]
   1:kernel-ec2                     ###########################################
[100%]

Kernel image:    /boot/vmlinux-2.6.35-rc4-8-ec2
Initrd image:    /boot/initrd-2.6.35-rc4-8-ec2
Root device:     /dev/sda1 (mounted on / as ext3)

Features:        block
14807 blocks
```

2. Generate an initrd on your running Linux instance.

```
# mkinitrd
Kernel image:    /boot/vmlinux-2.6.35-rc4-8-ec2
Initrd image:    /boot/initrd-2.6.35-rc4-8-ec2
Root device:     /dev/sda1 (mounted on / as ext3)
Features:        block
14806 blocks
```

3. Populate /boot/grub/menu.lst referencing your kernel on your running Linux instance.

Important

Your must modify your own menu.lst for your specific environment.

```
default 0
timeout 3

title EC2
```

```
        root (hd0)
        kernel /boot/vmlinux-ec2 root=/dev/sda1
        initrd /boot/initrd-ec2
```

4. Select an appropriate AKI ID from the Amazon Kernel Image IDs section that follows. For this host, we've chosen aki-88aa75e1 because we are bundling an AMI. You must not specify an Amazon ramdisk image (ARI) when using a PV-GRUB AKI.

5. For new AMIs or Amazon EBS volumes, bundle the AMI and set the default to your chosen AKI from your running Linux instance.

```
# ec2-bundle-vol -r x86_64 -d /mnt -p openSUSE-11.2-PV-GRUB -u [AWS-ID] -k
 /mnt/pkey.pem -c /mnt/cert.pem -s 10240 -e /mnt,/root/.ssh  --kernel aki-
88aa75e1
```

6. Upload the bundle from your running Linux instance to Amazon S3.

```
# ec2-upload-bundle -b MyReallyCoolBucketLocation -m /mnt/openSUSE-11.2-PV
GRUB.manifest.xml -a MyAccessKey -s MySecretKey
```

7. Register the AMI with the AKI (aki-88aa75e1) from your desktop using the Amazon EC2 command line tools.

```
$ ec2-register --name openSUSE-11.2-PVGRUB MyReallyCoolBucketLoca
tion/openSUSE-11.2-PVGRUB.manifest.xml
```

Amazon Support for PV-GRUB

Amazon supports the use of PV-GRUB to load a kernel of your choice for your AMI. However, we cannot provide support for your kernel itself or failures caused by the use of a kernel that does not meet the requirements of PV-GRUB. To avoid the situation in which a kernel does not work consistently or at all, we recommend that you use a known good kernel, select a non-PV-GRUB AKI, or seek support from your AMI vendor. Unfortunately, due to the wide and varied kernel landscape, it is impossible for Amazon to provide support for all kernel varieties.

Technical Notes for Advanced Users

The following information is provided to assist those who are familiar with compiling kernels. Many Linux distributions provide documentation on how to compile a kernel in kernel source packages. For users compiling 2.6.30+ kernels, Amazon recommends using the PVOps method described below. Amazon is unable to offer support for compiling your own kernel.

openSUSE and Gentoo Kernels

Both openSUSE and Gentoo provide native source code that enables compiling an Amazon EC2 capable kernel. For openSUSE, the configuration options are part of the mainline 11.2 and higher, while Gentoo provides a Xen-compatible in the "xen-sources" portage package. For both distributions, please use the following kernel configuration options:

CONFIG_XEN_COMPAT_00002_AND_LATER=y
CONFIG_XEN_COMPAT=0x030002
CONFIG_HOTPLUG_CPU=y

Option 1: Patching vanilla or other distribution kernels (apply Xen DomU patchset)

The process of patching any kernel (vanilla or distribution) is an advanced topic. For instructions, see "kernel/Documentation/applying-patches.txt" under the kernel source tree for instructions.

The same patchset that is used with openSUSE and Gentoo can be applied to a vanilla kernel.org kernel.

http://code.google.com/p/gentoo-xen-kernel/downloads/list
2.6.29: http://lists.xensource.com/archives/html/xen-users/2009-04/msg00004.html
http://lists.xensource.com/archives/html/xen-devel/2009-06/msg00127.html
2.6.30: http://lists.xensource.com/archives/html/xen-devel/2009-07/msg00027.html
2.6.31: http://lists.xensource.com/archives/html/xen-users/2009-10/msg00342.html
2.6.31-9: http://lists.xensource.com/archives/html/xen-users/2009-12/msg00411.html
2.6.31-10: http://lists.xensource.com/archives/html/xen-users/2010-01/msg00032.html
2.6.31-14 and 2.6.32-1 :http://lists.xensource.com/archives/html/xen-users/2010-04/msg00091.html

Option 2: Patching vanilla or other distribution kernels (PVOps)

For PVOps kernels, you can elect to disable the XSAVE hypercall in the guest. The following patch works against 2.6.32 through 2.6.35 kernels.

```
---
 arch/x86/xen/enlighten.c |    1 +
 1 files changed, 1 insertions(+), 0 deletions(-)

diff --git a/arch/x86/xen/enlighten.c b/arch/x86/xen/enlighten.c
index 52f8e19..6db3d67 100644
--- a/arch/x86/xen/enlighten.c
+++ b/arch/x86/xen/enlighten.c
@@ -802,6 +802,7 @@ static void xen_write_cr4(unsigned long cr4)
 {
  cr4 &= ~X86_CR4_PGE;
  cr4 &= ~X86_CR4_PSE;
+ cr4 &= ~X86_CR4_OSXSAVE;

  native_write_cr4(cr4);
 }
--
1.6.6.1
```

Amazon EC2 Instances

Topics

This section provides information about instances and their basic characteristics. It also describes how to launch an instance and connect to it.

Amazon EC2 provides templates known as Amazon Machine Images (AMIs) that contain pre-configured software such as an operating system, application server, and applications. You use these templates to launch an instance, which is a running copy of an AMI. Your instance keeps running until you stop or terminate it.

Your AWS account has a limit on the number of running instances you can have. There's also a limit on the *overall* number of instances you can have with any status except `Terminated`. This overall instance limit is 2x the running instance limit. You can request to increase your running instance limit by completing the Amazon EC2 Instance Limit Request Form.

Related Topics

Instance Basics

Topics

This section provides information about key concepts to help you understand Amazon EC2 instances.

Instance Families and Types

Topics

Amazon Elastic Compute Cloud (Amazon EC2) instances are grouped into the general families described in the following table.

Family	Description
Micro	Provide a small amount of consistent CPU resources and enable you to burst CPU capacity when additional cycles are available. They're well suited for lower throughput applications and websites that consume significant compute cycles periodically. For more information, see Micro Instances (p. 90).
Standard	Have memory-to-CPU ratios suitable for most general-purpose applications.
High-CPU	Have proportionally more CPU resources than memory (RAM). They're well suited for compute-intensive applications.
High-Memory	Have proportionally more memory resources. They're well suited for high-throughput applications, such as database and memory caching applications.
High I/O	Provide tens of thousands of low-latency, random I/O operations per second (IOPS) to an application. They're well suited for NoSQL databases, clustered databases, and OLTP (online transaction processing) systems. For more information, see High I/O Instances (p. 97).
Cluster Compute	Have a very large amount of CPU coupled with increased networking performance. They're well suited for High Performance Compute (HPC) applications and other demanding network-bound applications. For more information, see Overview (p. 99).
Cluster GPU	Provide general-purpose graphics processing units (GPUs), with proportionally high CPU and increased network performance for applications that benefit from highly parallelized processing. They're well suited for HPC applications as well as rendering and media processing applications. For more information, see Overview (p. 99).

Tip

One of the advantages of Amazon EC2 is that you pay by the instance hour, which makes it convenient and inexpensive to test the performance of your application on different instance families and types. A good way to determine the most appropriate instance family and instance type is to launch test instances and benchmark your application.

Available Instance Types

When you launch an instance, you specify the instance type (the value in the *Name* column in the following table). We launch an m1.small if you don't specify a particular instance type.

Type	EC2 Compute Units (ECU)	Virtual Cores	Memory	Instance Store Volumes	Platform	I/O	Name
Micro	Up to 2 (for short periodic bursts)	1	615 MiB	None (use Amazon EBS volumes for storage)	32-bit and 64-bit	Low	t1.micro
Small	1	1	1.7 GiB	150 GiB (1 x 150 GiB)	32-bit and 64-bit	Moderate	m1.small
Medium	2	1	3.75 GiB	400 GiB (1 x 400 GiB)	32-bit and 64-bit	Moderate	m1.medium
Large	4	2 (with 2 ECUs each)	7.5 GiB	840 GiB (2 x 420 GiB)	64-bit	High	m1.large
Extra Large	8	4 (with 2 ECUs each)	15 GiB	1680 GB (4 x 420 GiB)	64-bit	High	m1.xlarge
High-CPU Medium	5	2 (with 2.5 ECUs each)	1.7 GiB	340 GiB (1 x 340 GiB)	32-bit and 64-bit	Moderate	c1.medium
High-CPU Extra Large	20	8 (with 2.5 ECUs each)	7 GiB	1680 GiB (4 x 420 GiB)	64-bit	High	c1.xlarge
High-Memory Extra Large	6.5	2 (with 3.25 ECUs each)	17.1 GiB	410 GiB (1 x 410 GiB)	64-bit	Moderate	m2.xlarge
High-Memory Double Extra Large	13	4 (with 3.25 ECUs each)	34.2 GiB	840 GiB (1 x 840 GiB)	64-bit	High	m2.2xlarge

Type	EC2 Compute Units (ECU)	Virtual Cores	Memory	Instance Store Volumes	Platform	I/O	Name
High-Memory Quadruple Extra Large	26	8 (with 3.25 ECUs each)	68.4 GiB	1680 GiB (2 x 840 GiB)	64-bit	High	m2.4xlarge
High I/O Quadruple Extra Large	35	8 (with 4.37 ECUs each)	60.5 GiB	2 TiB (2 x 1 TiB SSD)	64-bit	Very high (10 Gbps Ethernet)	hi1.4xlarge
Cluster Compute Quadruple Extra Large	33.5	8 (2 x Intel Xeon X5570, quad-core "Nehalem" architecture)	22.5 GiB	1690 GiB (2 x 840 GiB)	64-bit	Very high (10 Gbps Ethernet)	cc1.4xlarge
Cluster Compute Eight Extra Large	88	16 (2 x Intel Xeon E5-2670, eight-core "Sandy Bridge" architecture)	60.5 GiB	3360 GiB (4 x 840 GiB)	64-bit	Very high (10 Gbps Ethernet)	cc2.8xlarge
Cluster GPU Quadruple Extra Large	33.5	8 (2 x Intel Xeon X5570, quad-core "Nehalem" architecture), plus 2 NVIDIA Tesla M2050 "Fermi" GPUs	22.5 GiB (see note after this table)	1680 GiB (2 x 840 GiB)	64-bit	Very high (10 Gbps Ethernet)	cg1.4xlarge

Note

The hi1.4xlarge instance type is based on solid-state drive (SSD) technology. For more information, see High I/O Instances (p. 97).

Note

The cg1.x4large instance type has 1 GiB reserved for GPU operation. The 21.5 GiB doesn't include the on-board memory of the GPUs, which is 3 GiB per GPU for the NVIDIA Tesla M2050.

For more information about instance store volumes, see Instance Store Device Names (p. 481).

EBS-Optimized Instances

An EBS-Optimized instance uses an optimized configuration stack and provides additional, dedicated capacity for EBS I/O. This optimization provides the best performance for your EBS volumes by minimizing contention between EBS I/O and other traffic from your Amazon EC2 instance.

EBS optimization enables instances to fully utilize the IOPS provisioned on an EBS volume. EBS-Optimized instances deliver dedicated throughput to EBS, with options between 500 Mbps and 1,000 Mbps, depending on the instance type you use. When attached to an EBS-Optimized instance, Provisioned IOPS volumes are designed to deliver within 10 percent of their provisioned performance 99.9 percent of the time in a given year. For more information, see Provisioned IOPS Volumes (p. 436).

The following instance types can be launched as EBS-Optimized instances:

*   Large (m1.large)
*   Extra Large (m1.xlarge)
*   High-Memory Quadruple Extra Large (m2.4xlarge)

When you use an EBS-Optimized instance, you pay an additional low, hourly fee for the dedicated capacity.

Private IP Addresses Per ENI Per Instance Type

The following table lists the maximum number of Elastic Network Interfaces (ENIs) per EC2 instance type and the maximum number of private IP addresses per ENI. ENIs and multiple private IP addresses are only available in EC2 instances running in Amazon Virtual Private Cloud. For more information, see Multiple IP Addresses (p. 386).

Type	Elastic Network Interfaces (ENIs)	Private IP Addresses per ENI	Name
Micro	N/A	N/A	t1.micro
Small	2	4	m1.small
Medium	2	6	m1.medium
Large	3	10	m1.large
Extra Large	4	15	m1.xlarge
High-CPU Medium	2	6	c1.medium
High-CPU Extra Large	4	15	c1.xlarge
High-Memory Extra Large	4	15	m2.xlarge
High-Memory Double Extra Large	4	30	m2.2xlarge
High-Memory Quadruple Extra Large	8	30	m2.4xlarge
High I/O Quadruple Extra Large	8	30	hi1.4xlarge
Cluster Compute Quadruple Extra Large	N/A	N/A	cc1.4xlarge

Type	Elastic Network Interfaces (ENIs)	Private IP Addresses per ENI	Name
Cluster Compute Eight Extra Large	8	30	cc2.8xlarge
Cluster GPU Quadruple Extra Large	N/A	N/A	cg1.4xlarge

Windows Instance Types

Amazon EC2 instances can run Microsoft Windows Server 2003, Microsoft Windows Server 2008 or Microsoft Windows Server 2008 R2. The Windows AMIs provide you with all standard Microsoft Windows Server functionality.

Using Amazon EC2 instances running Windows is similar to using instances running Linux/UNIX. The following are the major differences between instances that use Linux/UNIX and Windows:

- **Remote Desktop—**To access Windows instances, you use Remote Desktop instead of SSH.
- **Administrative Password—**To access Windows instances the first time, you must obtain the administrative password (available through the AWS Management Console, the command line tools, or the EC2 API).
- **Bundling—**Amazon instance store-backed Windows instances requires different bundling procedures than Amazon instance store-backed Linux/UNIX instances. For more information, go to Creating an Instance Store-Backed Windows AMI in the *Amazon EC2 Microsoft Windows Guide* .

Amazon EC2 currently provides the following Windows AMIs:

- Microsoft Windows Server 2003 (32-bit)
- Microsoft Windows Server 2003 (64-bit)
- Microsoft Windows Server 2008 (32-bit)
- Microsoft Windows Server 2008 (64-bit)
- Microsoft Windows Server 2008 R2 (64-bit)

The Windows public AMIs that Amazon provides are unmodified versions of Windows with the following two exceptions: we added drivers to improve the networking and disk I/O performance and we created the Amazon EC2 configuration service. The Amazon EC2 configuration service performs the following functions:

- Randomly sets the Administrator password on initial launch, encrypts the password with the user's SSH key, and reports it to the console. This operation happens upon initial AMI launch. If you change the password, AMIs that are created from this instance use the new password.
- Configures the computer name to the internal DNS name. To determine the internal DNS name, see Instance IP Addresses (p. 383).
- Sends the last three system and application errors from the event log to the console. This helps developers to identify problems that caused an instance to crash or network connectivity to be lost.

For more information about the EC2 configuration service, see ???.

Compute Resources Measurement

Changing 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. One EC2 Compute Unit provides the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor. This is also the equivalent to an early-2006 1.7 GHz Xeon processor referenced in our original documentation.

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, go to the Instance page.

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

I/O Resources

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

The different instance types 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 (low, moderate, high, etc.). 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 higher I/O performance.

Instance Tags

To help categorize and manage your instances, you can assign *tags* of your choice to them. For more information, see Tagging Your Resources (p. 508).

Micro Instances

Topics

- Optimal Application of Micro Instances (p. 91)
- Available CPU Resources During Spikes (p. 93)
- When the Instance Uses Its Allotted Resources (p. 93)
- Comparison with the m1.small Instance Type (p. 95)
- AMI Optimization for Micro Instances (p. 97)
- Related Topics (p. 97)

Micro instances (t1.micro) provide a small amount of consistent CPU resources and allow you to increase CPU capacity in short bursts when additional cycles are available. They are well suited for lower throughput applications and web sites that require additional compute cycles periodically.

This section describes how micro instances generally work so you can understand how to apply them. Our intent is not to specify exact behavior, but to give you visibility into the instance's behavior so you can understand its performance (to a greater degree than you typically get with, for example, a multi-tenant shared web hosting system).

The micro instance is available on both 32-bit and 64-bit platforms, but it's available as an Amazon EBS-backed instance only. For basic specifications for the micro instance type, see Available Instance Types (p. 86).

Optimal Application of Micro Instances

Micro instances provide spiky CPU resources for workloads that have a CPU usage profile similar to what is shown in the following figure.
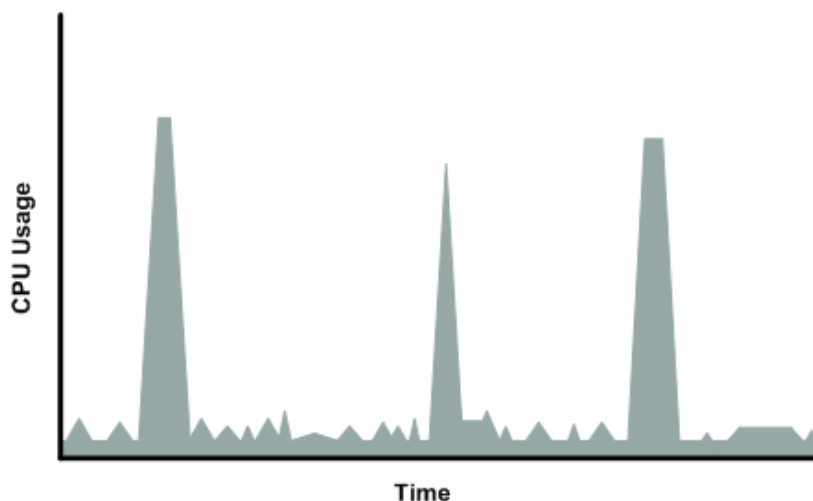
The instance is designed to operate with its CPU usage at essentially only two levels: the normal low background level, and then at brief spiked levels much higher than the background level. We allow the instance to operate at up to 2 EC2 compute units (ECUs) (for more information, see Compute Resources Measurement (p. 90)). The ratio between the maximum level and the background level is designed to be large. Micro instances are designed to support tens of requests per minute on your application. However, actual performance can vary significantly depending on the amount of CPU resources required for each request on your application.

Your application might have a different CPU usage profile than that described in the preceding section. The next figure shows the profile for an application that isn't appropriate for a micro instance. The application requires continuous data-crunching CPU resources for each request, resulting in plateaus of CPU usage that the micro instance isn't designed to handle.
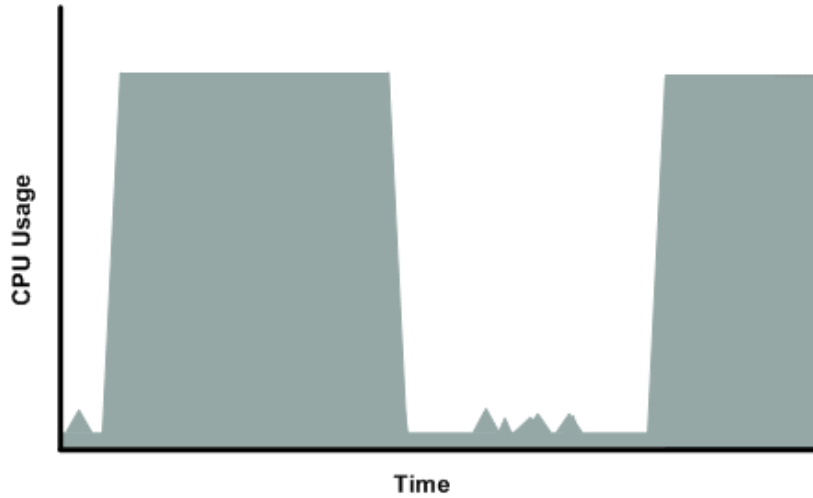
The next figure shows another profile that isn't appropriate for a micro instance. Here the spikes in CPU use are brief, but they occur too frequently to be serviced by a micro instance.
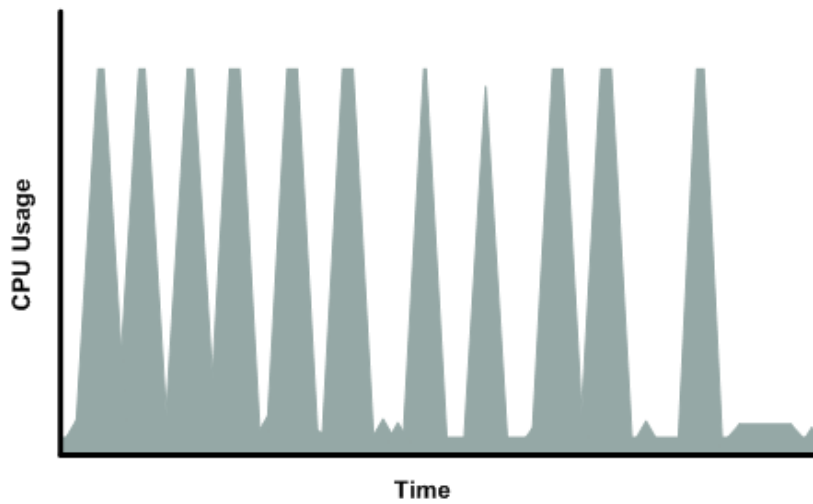
The next figure shows another profile that isn't appropriate for a micro instance. Here the spikes aren't too frequent, but the background level between spikes is too high to be serviced by a micro instance.
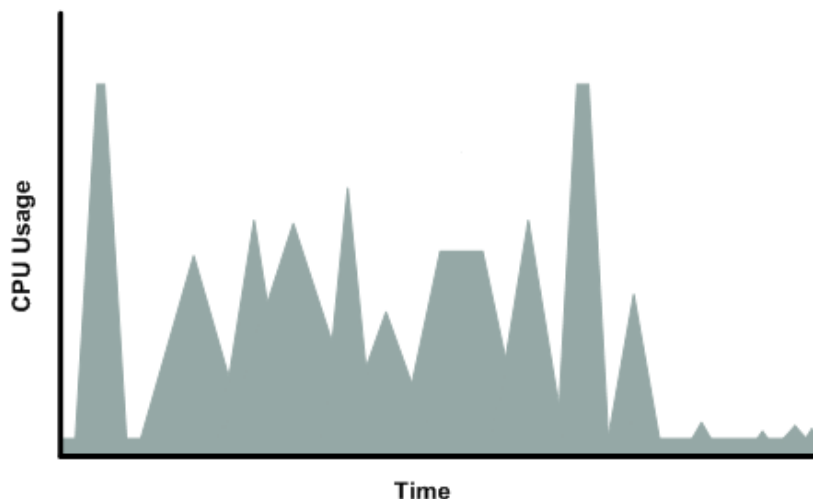
In each of the preceding cases of workloads not appropriate for a micro instance, we recommend you consider using a different instance type (for more information about the instance types, see Instance Families and Types (p. 85)).

Available CPU Resources During Spikes

When your instance *bursts* to accommodate a spike in demand for compute resources, it uses unused resources on the host. The amount available depends on how much contention there is when the spike occurs. The instance is never left with no CPU resources, whether other instances on the host are spiking or not.

When the Instance Uses Its Allotted Resources

We expect your application to consume only a certain amount of CPU resources in a period of time. If the application consumes more than your instance's allotted CPU resources, we temporarily limit the instance so it operates at a low CPU level. If your instance continues to use all of its allotted resources, its performance will degrade. We will increase the time we limit its CPU level, thus increasing the time before the instance is allowed to burst again.

If you enable Amazon CloudWatch monitoring for your micro instance, you can use the "Avg CPU Utilization" graph in the AWS Management Console to determine whether your instance is regularly using all its allotted CPU resources. We recommend that you look at the maximum value reached during each given period. If the maximum value is 100%, we recommend that you use Auto Scaling to scale out (with additional micro instances and a load balancer), or move to a larger instance type. For more information about Auto Scaling, go to the Auto Scaling Developer Guide.

Note

Micro instances are available as either 32-bit or 64-bit. If you need to move to a larger instance, make sure to move to a compatible instance type.

The following figures show the three suboptimal profiles from the preceding section and what it might look like when the instance consumes its allotted resources and we have to limit its CPU level. If the instance consumes its allotted resources, we restrict it to the low background level.

The next figure shows the situation with the long plateaus of data-crunching CPU usage. The CPU hits the maximum allowed level and stays there until the instance's allotted resources are consumed for the period. At that point, we limit the instance to operate at the low background level, and it operates there

until we allow it to burst above that level again. The instance again stays there until the allotted resources are consumed and we limit it again (not seen on the graph).
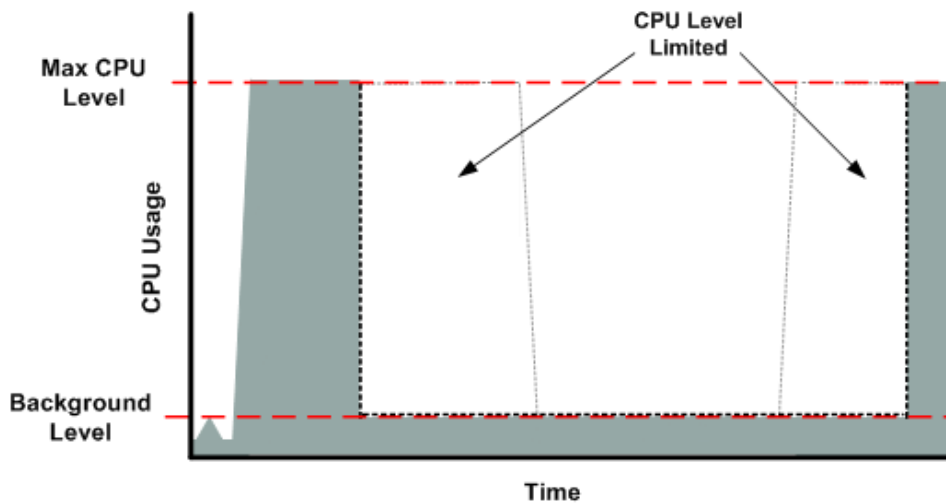
The next figure shows the situation where the requests are too frequent. The instance uses its allotted resources after only a few requests and so we limit it. After we lift the restriction, the instance maxes out its CPU usage trying to keep up with the requests, and we limit it again.
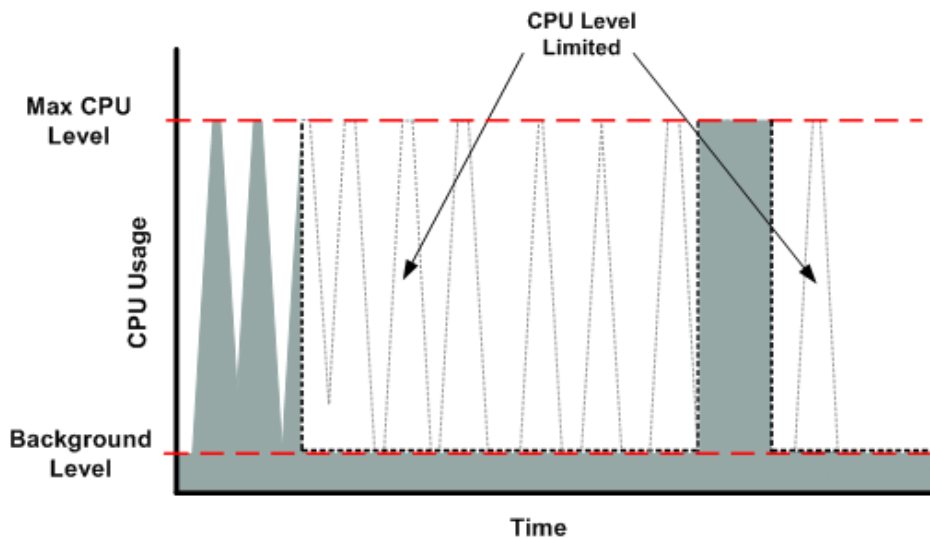
The next figure shows the situation where the background level is too high. Notice that the instance doesn't have to be operating at the maximum CPU level for us to limit it. We limit the instance when it's operating above the normal background level and has consumed its allotted resources for the given period. In this case (as in the preceding one), the instance can't keep up with the work, and we limit it again.
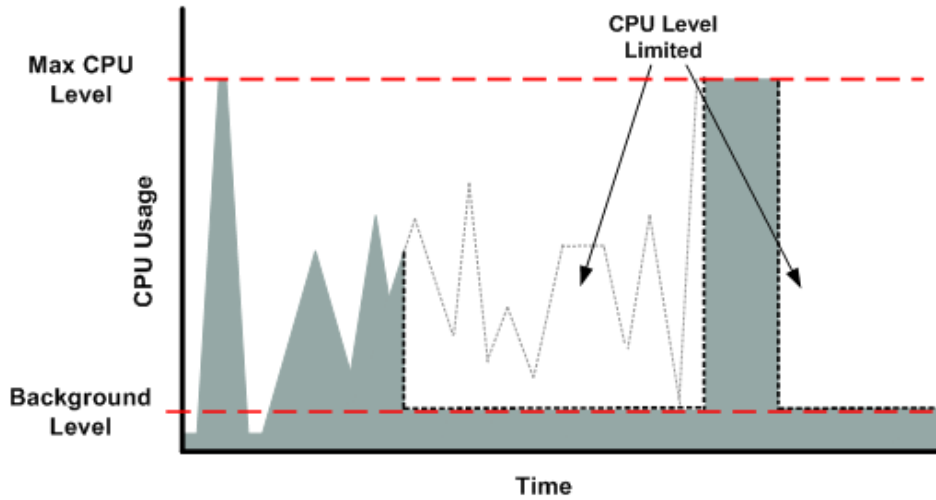
Comparison with the m1.small Instance Type

The micro instance provides different levels of CPU resources at different times (up to 2 ECUs). By comparison, the m1.small instance type provides 1 ECU at all times. The following figure illustrates the difference.
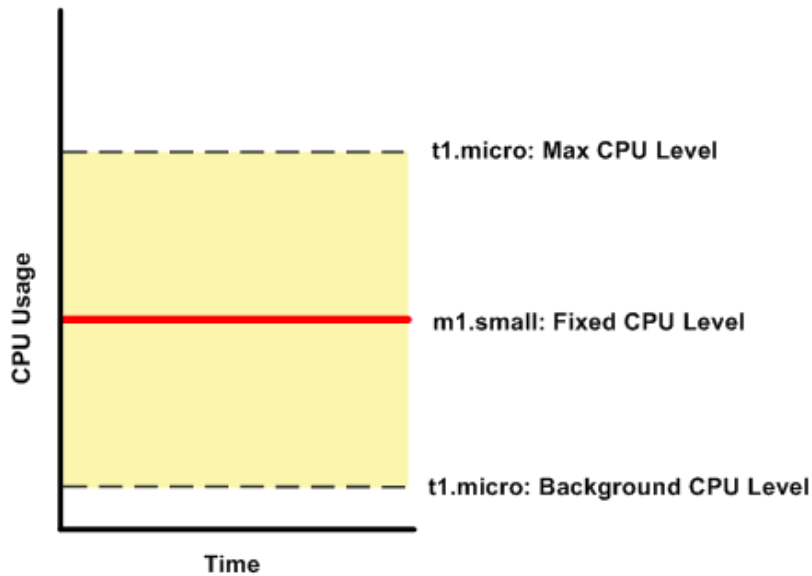
The following figures compare the CPU usage of a micro instance with an m1.small for the various scenarios we've discussed in the preceding sections.

The first figure that follows shows an optimal scenario for a micro instance (the left graph) and how it might look for an m1.small (the right graph). In this case, we don't need to limit the micro instance. The processing time on the m1.small would be longer for each spike in CPU demand compared to the micro instance.
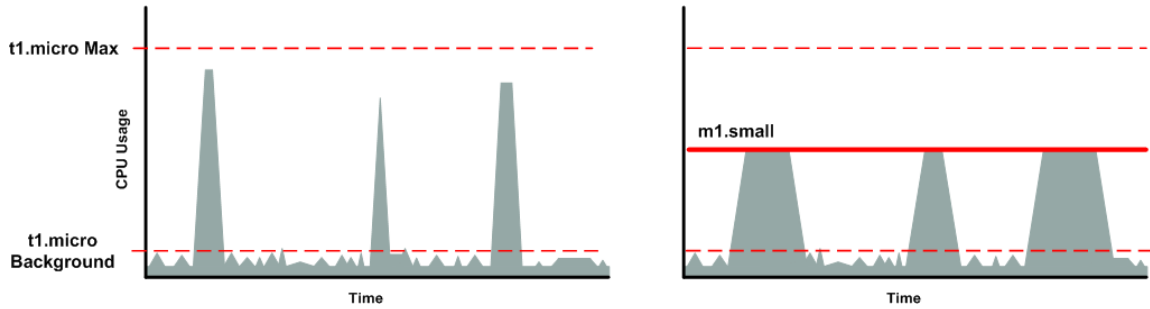
The next figure shows the scenario with the data-crunching requests that used up the allotted resources on the micro instance, and how they might look with the m1.small instance.
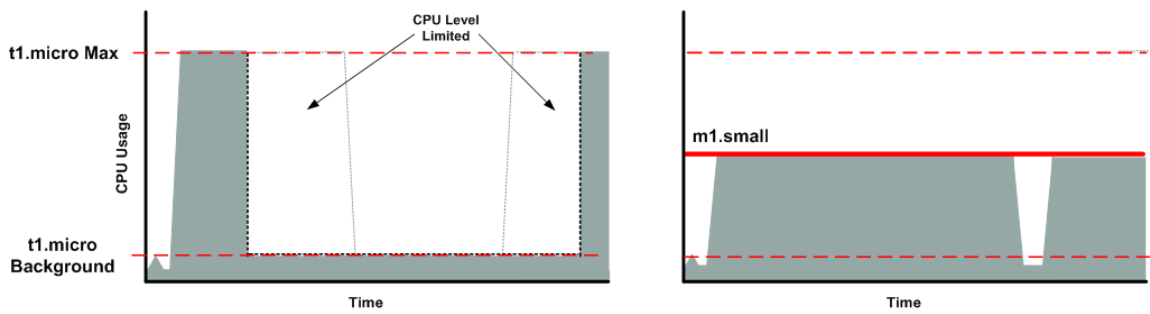
The next figure shows the frequent requests that used up the allotted resources on the micro instance, and how they might look on the m1.small instance.
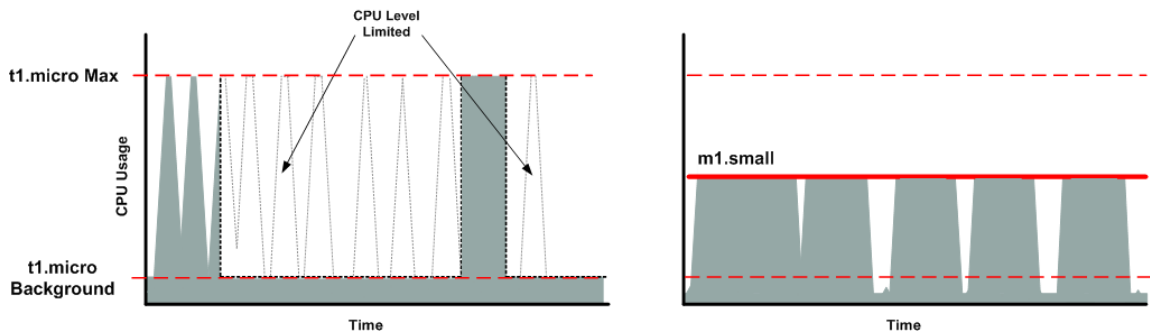
The next figure shows the situation where the background level used up the allotted resources on the micro instance, and how it might look on the m1.small instance.
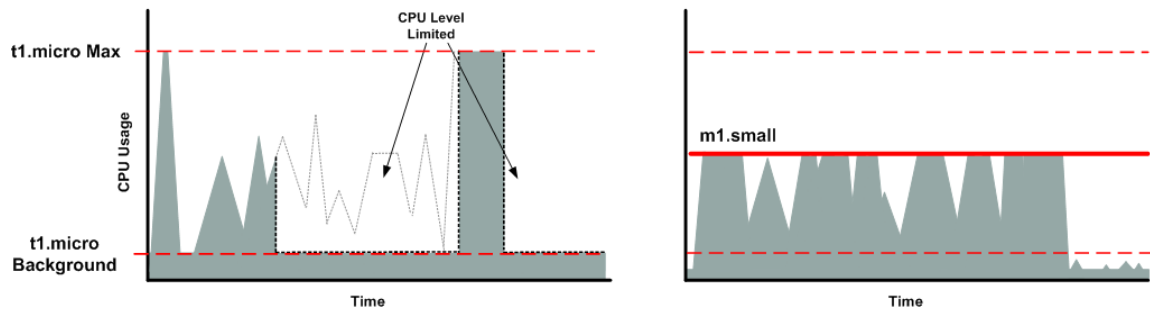
AMI Optimization for Micro Instances

We recommend you follow these best practices when optimizing an AMI for the micro instance type:

- Design the AMI to run on 600 MB of RAM
- Limit the number of recurring processes that use CPU time (e.g., cron jobs, daemons)

In Linux, you can optimize performance using swap space and virtual memory (e.g., by setting up swap space in a separate partition from the root file system).

In Windows, when you perform significant AMI or instance configuration changes (e.g., enable server roles or install large applications), you might see limited instance performance, because these changes can be memory intensive and require long-running CPU resources. We recommend you first use a larger instance type when performing these changes to the AMI, and then run the AMI on a micro instance for normal operations.

Related Topics

- Instance Families and Types (p. 85)

High I/O Instances

High I/O instances (hi1.4xlarge) can deliver tens of thousands of low-latency, random I/O operations per second (IOPS) to applications. They are well suited for the following scenarios:

- NoSQL databases (for example, Cassandra and MongoDB)
- Clustered databases
- Online transaction processing (OLTP) systems

By default, you can run up to two high I/O instances. If you need more than two high I/O instances, you can request more using the Amazon EC2 Instance Request Form.

You can cluster high I/O instances in a cluster placement group. The placement group cannot contain other cluster instances.

Hardware Specifications

The hi1.4xlarge instance type is based on solid-state drive (SSD) technology.

The hardware specifications for hi1.4xlarge instances are described in the following table.

Item	Description
Processor	35 EC2 compute units
Virtual cores	16 virtual cores (8 cores + 8 hyperthreads)
Memory	60.5 GiB
Platform	64-bit
Network	10 Gbps Ethernet
Instance store volumes	2 x 1 TiB (SSD)

Disk I/O Performance

Using Linux paravirtual (PV) AMIs, high I/O instances can deliver more than 120,000 4 KB random read IOPS and between 10,000 and 85,000 4 KB random write IOPS (depending on active logical block addressing span) to applications across two 1 TiB data volumes. For hardware virtual machines (HVM) and Windows AMIs, performance is approximately 90,000 4 KB random read IOPS and between 9,000 and 75,000 4 KB random write IOPS.

The maximum sequential throughput on all AMI types (Linux PV, Linux HVM, and Windows) per second is approximately 2 GB read and 1.1 GB write.

SSD Storage

This section contains important information you need to know about SSD storage. With SSD storage:

- The primary data source is an instance store with SSD storage.
- Read performance is consistent and write performance can vary.
- Write amplification can occur.
- The TRIM command is not currently supported.

Instance Store with SSD Storage

High I/O instances use an EBS-backed root device. However, their primary data storage is provided by the SSD volumes in the instance store. Like other instance store volumes, these instance store volumes persist only for the life of the instance. Because the root device of the high I/O instance is EBS-backed, you can still start and stop your instance. When you stop an instance, your application persists, but your production data in the instance store does not persist. For more information about instance store volumes, see Amazon EC2 Instance Store (p. 479).

Variable Write Performance

Write performance depends on how your applications utilize logical block addressing (LBA) space. If your applications use the total LBA space, write performance can degrade by about 90 percent. Benchmark your applications and monitor the queue depth (the number of pending I/O requests for a volume) and I/O size.

Write Amplification

Write amplification refers to an undesirable condition associated with flash memory and SSDs, where the actual amount of physical information written is a multiple of the logical amount intended to be written. Because flash memory must be erased before it can be rewritten, the process to perform these operations results in moving (or rewriting) user data and metadata more than once. This multiplying effect increases

the number of writes required over the life of the SSD, which shortens the time that it can reliably operate. High I/O instances are designed with a provisioning model intended to minimize write amplification.

Random writes have a much more severe impact on write amplification than serial writes. If you are concerned about write amplification, allocate less than the full tebibyte of storage for your application (also known as over provisioning).

The TRIM Command

The TRIM command enables the operating system to notify an SSD that blocks of previously saved data are considered no longer in use. TRIM limits the impact of write amplification.

TRIM support is not available for high I/O instances at this time. We hope to add TRIM support in the future.

Related Topics

- Instance Families and Types (p. 85)

Cluster Instances

Topics

In this topic, you'll learn about cluster instances and how to create hardware virtual machine (HVM) AMIs. You'll also learn how to create placement groups and then launch cluster instances into them.

Overview

Amazon EC2 offers two *cluster instance types*: cluster compute instances and cluster graphics processing unit (GPU) instances. These instance types provide a very large amount of computational power coupled with increased network performance. They are well suited for High Performance Compute (HPC) applications and other demanding network-bound applications, such as many science and engineering applications, financial modeling applications, and business analytics applications. You can logically group cluster instances into clusters (known as *cluster placement groups*). This allows applications to get the full-bisection bandwidth and low-latency network performance required for tightly coupled, node-to-node communication typical of many HPC applications.

Cluster GPU instances have the same networking and placement group properties as cluster compute instances, but also include NVIDIA Tesla M2050 "Fermi" GPUs. These GPUs can be used to accelerate many scientific, engineering, and rendering applications by leveraging the Compute Unified Device Architecture (CUDA) or OpenCL parallel computing frameworks.

Cluster instances are an Amazon EC2 instance type. This means that cluster instances interoperate with other Amazon EC2 features and AWS cloud services just as other Amazon EC2 instance types do. The functionality of cluster instances is similar to the functionality of other Amazon EC2 instance types, but with the important differences described in the following table and sections.

Feature	Description
Cluster Placement Groups	Provides low latency and high-bandwidth connectivity between the cluster instances within a single Availability Zone. You just create a cluster placement group and then launch the cluster instances into it. Instances launched within a placement group participate in a full-bisection bandwidth cluster appropriate for HPC applications. For information about creating cluster placement groups, see Creating a Cluster Placement Group (p. 102).
HVM-Based Virtualization	Cluster instances run as hardware virtual machine (HVM)-based instances. HVM virtualization uses hardware-assist technology provided by the AWS platform. With HVM virtualization the guest VM runs as if it were on a native hardware platform, except that it still uses paravirtual (PV) network and storage drivers for improved performance. For more information, see HVM-Based AMIs (p. 100).
Amazon EBS-Backed, HVM-Based AMIs	You must launch cluster instances from Amazon EBS-backed, HVM-based AMIs. Launching an instance from Amazon EC2 instance store-backed AMIs is not supported. AWS labels all AMIs as either `hvm` or `paravirtual` so you can determine the virtualization type. For information about using Amazon EBS-backed AMIs, see Storage for the Root Device (p. 24).

Current Limitations

Cluster instances currently have the following limitation:

- Cluster instance types are not available in every Region. For more information, see the Amazon EC2 Release Notes.
- They are not available to use with Amazon DevPay.

Cluster placement groups currently have the following limitations:

- Amazon EC2 Reserved Instances are not guaranteed within specific requested cluster placement group.
- A cluster placement group cannot span multiple Availabilty Zones.

HVM-Based AMIs

Cluster instances require the use of HVM-based, Amazon EBS-backed AMIs. Paravirtual (PV) AMIs employed by Linux instances and other EC2 instance types that use paravirtual machine (PVM)-based virtualization are not supported. Cluster instances do not require the use of Amazon Kernel Images (AKIs) or Amazon RAM disk Images (ARIs) that are part of PVM-based AMIs used with other EC2 instance types.

Reference AMIs

To help you get started with cluster instances, AWS provides a reference Amazon Linux AMI (ami-321eed5b) and a reference Amazon Windows AMI (ami-a03ee2c9). The AMIs include the NVIDIA driver and CUDA toolkit, which enable the NVIDIA GPUs. To learn more about the Amazon Linux reference AMI, go to Amazon Linux AMI. For information on how to use the Amazon Linux reference AMI, see Amazon Linux AMIs (p. 70). To learn more about the Amazon Windows AMIs, go to Amazon Machine Images (AMIs). For information on how to use the Amazon Windows reference AMI, see Amazon Windows AMI Basics. You can customize an instance and create your own AMI from it (for more information, see Creating an HVM AMI from a Running Instance (p. 101)).

How to Create and Launch a Cluster Instance

You can use the following steps to create, customize, and launch a cluster instance.

1. Launch a cluster instance. See Launching a Cluster Instance (p. 101).
2. Optimize your instance by putting the software you need on it. See Optimizing Your Cluster Instance (p. 101).
3. Create an HVM AMI from a running instance. See Creating an HVM AMI from a Running Instance (p. 101).
4. Create a cluster placement group. See Creating a Cluster Placement Group (p. 102).
5. Launch an cluster instance into a cluster placement group. See Launching a Cluster Instance into a Cluster Placement Group (p. 104).
6. (Optional) Use AWS CloudFormation to automate these tasks for you. See AWS CloudFormation Templates for Cluster Instances (p. 107).

Launching a Cluster Instance

You can launch a cluster compute or cluster GPU instance just as you would launch any other type of instance. For more information on how to launch an instance, see Launching Amazon EC2 Instances (p. 217).

Optimizing Your Cluster Instance

The process for optimizing your cluster compute or cluster GPU instance is the same as with any other type of instance. First, install and configure your software on the instance. When you're through, you're ready to create an AMI. For more information, see Creating Your Own AMIs (p. 35).

Creating an HVM AMI from a Running Instance

To launch cluster instances in Amazon EC2, you must use an Amazon EBS-backed HVM AMI. These AMIs have been designed specifically for cluster computing.

You can create an HVM AMI by customizing a running instance of another HVM AMI, and then creating an image from that instance. We have provided an Amazon Linux HVM-based AMI and an Amazon Windows HVM-based AMI for you to use as a starting point for your customization. You can use the AWS Management Console, the command line interface (CLI), or the application programming interface (API) to customize your HVM AMI.

AWS Management Console

To create an HVM AMI from a running instance

1. For instructions that use the AWS Management Console, see Creating Amazon EBS-Backed AMIs Using the Console (p. 33).
2. Use the reference Cluster GPU Amazon Linux AMI (ami-321eed5b) or the Microsoft Windows 2008 R2 64-bit for Cluster Instances AMI (ami-a03ee2c9).

CLI

To create an HVM AMI from a running instance

1. Use `ec2-create-image` to create your own image from your customized instance.

   ```
   PROMPT>  ec2-create-image -n your_image_name  instance_id
   ```

For example:

```
PROMPT>  ec2-create-image -n "My_Custom_HVM_AMI" i-1a2b3c4d
```

Amazon EC2 begins creating the image and returns the new image's ID. For example:

```
IMAGE ami-5f4d3a2b
```

2. To check whether the AMI is ready, use the following command.

```
PROMPT>  ec2-describe-images -o self
```

Amazon EC2 returns information about the AMI.

API

To create an HVM AMI from a running instance

*   Construct a Query request similar to the following example:

```
https://ec2.amazonaws.com/
?Action=CreateImage
&InstanceId=i-1a2b3c4d
&Name=My_Custom_HVM_AMI
&AUTHPARAMS...
```

It returns a response similar to the following example:

```
<CreateImageResponse xmlns="http://ec2.amazonaws.com/doc/2012-07-20/">
  <imageId>ami-5f4d3a2b</imageId>
</CreateImageResponse>
```

Creating a Cluster Placement Group

A *cluster placement group* is a logical grouping of cluster instances within a single Availability Zone. A cluster placement group cannot span multiple Availability Zones. You first create a cluster placement group and then launch multiple cluster instances into the group. You must give each placement group a name that is unique within your account. Cluster compute instances are not available in every Region. For more information about cluster placement groups, see Overview (p. 99).

You can create a cluster placement group using the AWS Management Console, the command line interface (CLI), or the application programming interface (API).

Note

You can't merge cluster placement groups. Instead you must terminate the instances in one of the groups, and then relaunch the instances into the other group.

AWS Management Console

To create a cluster placement group

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

2. In the **Navigation** pane, click **Placement Groups**.

3. Click **Create Placement Group**.

4. In the **Placement Group** dialog box, provide a unique name for the placement group and click **Yes, Create**.

    Amazon EC2 creates the placement group.

CLI

To create a cluster placement group

1. Use the `ec2-create-placement-group` command. You must name the cluster placement group and set the `-s` option to `cluster`, as shown in the following example.

    ```
    PROMPT> ec2-create-placement-group XYZ-cluster -s cluster
    ```

    It returns a response similar to the following example:

    ```
    PLACEMENTGROUP  XYZ_cluster  cluster  available
    ```

2. Record the name of the placement group. You will need this name when you launch your cluster instances into this group.

3. Use the `ec2-describe-placement-groups` command to get details about your cluster placement groups.

    If you do not specify the name of the cluster placement group, details about all the placement groups in your account are returned.

    ```
    PROMPT> ec2-describe-placement-groups
    ```

    It returns a response similar to the following example:

    ```
    PLACEMENTGROUP  XYZ-cluster  cluster  available
    PLACEMENTGROUP  ABC-cluster  cluster  available
    ```

API

To create a cluster placement group

1. Construct a Query request, as shown in the following example. You must name the cluster placement group and specify the strategy as `cluster`.

    ```
    Action=CreatePlacementGroup
    &GroupName=XYZ-cluster
    ```

```
&Strategy=cluster
&AUTHPARAMS...
```

It returns a response similar to the following example:

```
<CreatePlacementGroupResponse xmlns="http://ec2.amazonaws.com/doc/2012-07-
20/">
    <requestID>d4904fd9-82c2-4ea5-adfe-a9cc3EXAMPLE</requestId>
    <return>true</return>
</CreatePlacementGroupResponse>
```

2. To get information about your cluster placement groups, construct a Query request similar to the following.

```
Action=DescribePlacementGroups
&GroupName.1=XYZ-cluster
&AUTHPARAMS...
```

It returns a response similar to the following example:

```
<DescribePlacementGroupsResponse xmlns="http://ec2.amazonaws.com/doc/2012-
07-20/">
   <requestId>d4904fd9-82c2-4ea5-adfe-a9cc3EXAMPLE</requestId>
      <placementGroupSet>
         <item>
             <groupName>XYZ-cluster</groupName>
             <strategy>cluster</strategy>
             <state>available</state>
         </item>
      </placementGroupSet>
</DescribePlacementGroupsResponse>
```

Launching a Cluster Instance into a Cluster Placement Group

Before you launch a cluster instance into a cluster placement group, ensure that you have the following:

*   A valid HVM AMI
*   An existing cluster placement group
*   An estimate of the total number of instances you want to launch in the placement group

You can launch a cluster instance into a cluster placement group using the AWS Management Console, the command line interface (CLI), or the application programming interface (API).

Important

We recommend that you launch the minimum number of instances that you need in the placement group at once in a single launch request. If you launch only a few instances and then try adding more instances to the placement group later, you increase your chances of getting an "Insufficient Capacity" error.

For instructions on how to create an HVM AMI, see Overview (p. 99). For instructions on how to create a cluster placement group, see Creating a Cluster Placement Group (p. 102).

For information about cluster placement groups, see Overview (p. 99).

AWS Management Console

Launching a cluster instance using the AWS Management Console is just like launching any other type of instance using the Classic Wizard. The only difference is that the Request Instances Wizard prompts you on whether you want to launch your instances into a placement group.

CLI

To launch instances in a cluster placement group

*   Use the `ec2-run-instances` command with the name of the placement group.

    The following example launches 10 cc2.8xlarge instances into the XYZ-cluster placement group.

    ```
    PROMPT> ec2-run-instances ami-7ea24a17 --type cc2.8xlarge –n 10 --placement-
    group XYZ-cluster
    ```

    It returns a response similar to the following example:

    ```
    RESERVATION     r-064bcd6d      999988887777    default
    INSTANCE i-89fea78c ... cc2.8xlarge ... us-east-1a ... XYZ-cluster   hvm
    ...
    ```

API

To launch instances in a placement group

*   Construct a Query request similar to the following example. This example launches 10 cc2.8xlarge instances in the XYZ-cluster placement group:

    ```
    Action=RunInstances
    &InstanceType=cc2.8xlarge
    &Placement.GroupName=XYZ-cluster
    &MinCount=10
    &MaxCount=10
    &AUTHPARAMS...
    ```

    It returns a response similar to the following example:

    ```
    <RunInstancesResponse xmlns="http://ec2.amazonaws.com/doc/2012-07-20/">
       ...
       <placement>
          <availabiltyZone>us-east-1a</availabilityZone>
          <groupName>XYZ-cluster</groupName>
       </placement>
       ...
       <virtualizationType>hvm</virtualizationType>
    </RunInstancesResponse>
    ```

Deleting a Cluster Placement Group

You might want to delete a cluster placement group if your cluster instance requirements change or if you no longer need the placement group.

You delete a cluster placement group using the AWS Management Console, the command line interface (CLI), or the application programming interface (API).

For information about cluster placement groups, see Overview (p. 99).

AWS Management Console

To delete a cluster placement group

Before you delete your cluster placement group, first you need to terminate all running instances in the group.

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the **Navigation** pane, click **Placement Groups**.
   The console displays a list of cluster placement groups that belong to the account.
3. Select the cluster placement group that you want to delete, and click **Delete**.
4. Click **Yes, Delete**.
   Amazon EC2 deletes the cluster placement group.

CLI

To delete a cluster placement group

Before you delete your cluster placement group, first you need to terminate all running instances in the group.

- After the running instances in the cluster placement group have been terminated, use the `ec2-delete-placement-group` command, as in the following example:

```
PROMPT> ec2-delete-placement-group XYZ-cluster
```

It returns a response similar to the following example:

```
PLACEMENTGROUP  XYZ-cluster  deleted
```

API

To delete a cluster placement group

Before you delete your cluster placement group, first you need to terminate all running instances in the group.

- After the running instances in the cluster placement group have been terminated, construct a Query request similar to the following example.

```
Action=DeletePlacementGroup
&GroupName=XYZ-cluster
&AUTHPARAMS...
```

It returns a response similar to the following example:

```
<DeletePlacementGroupResponse xmlns="http://ec2.amazonaws.com/doc/2012-07-
20/">
    <requestID>d4904fd9-82c2-4ea5-adfe-a9cc3EXAMPLE</requestId>
    <return>true</return>
</DeletePlacementGroupResponse>
```

AWS CloudFormation Templates for Cluster Instances

To make it easy for you to launch a cluster, you can use AWS CloudFormation templates that provision a two-node cluster with NFS storage, monitoring and a mounted 200 GB data set.

Each template performs the following tasks:

1. Creates a new controller instance.
2. Bootstraps MIT StarCluster. For more information about MIT StarCluster, see http://web.mit.edu/star/cluster/.
3. Generates a new cluster configuration.
4. Creates a new data volume.
5. Creates a new cluster master, worker instance, and storage.
6. Configures the resources to process compute jobs using Oracle Grid Engine.

You can use these templates to create a stack, log into the controller image, and use the cluster master to submit jobs or configure your cluster.

Here is where you can download the templates from AWS:

- Download the template for a cc2.8xlarge cluster from https://s3.amazonaws.com/cloudformation-templates-us-east-1/cc2-cluster.json
- Download the template for a cc1.4xlarge cluster from https://s3.amazonaws.com/cloudformation-templates-us-east-1/cc1-cluster.json

For more information about using AWS CloudFormation, see the AWS CloudFormation User Guide.

Using the NVIDIA Driver on a Cluster GPU Instance

We provide a reference Amazon Linux AMI (ami-321eed5b) and a reference Amazon Windows AMI (ami-a03ee2c9) that already have the NVIDIA driver installed. Amazon provides updated and compatible builds of the NVIDIA kernel drivers for each new official kernel upgrade. If you decide to use a different NVIDIA driver version than the one Amazon provides or, if you want to use a different kernel than an official Amazon Linux AMI build or an official Amazon Windows AMI build, then you will first need to uninstall the Amazon-provided NVIDIA packages from your system to avoid conflicts with the versions of the drivers you are trying to install.

Use this command to uninstall Amazon-provided NVIDIA packages:

```
[root@ip-xxx ~]# sudo yum erase nvidia cudatoolkit
```

Important

The Amazon-provided CUDA toolkit package has dependencies on the NVIDIA drivers.
Uninstalling the NVIDIA packages erases the CUDA toolkit. You will need to reinstall the CUDA
toolkit after installing the NVIDIA driver.

Manual Installation of the NVIDIA Driver

A cluster GPU instance must have the NVIDIA Tesla M2050 GPU driver. The NVIDIA driver you install
needs to be compiled against the kernel you intend to run on your instance.

Tip

You can download NVIDIA drivers at http://www.nvidia.com/Download/Find.aspx. Look for the
Tesla M-Class M2050 driver for Linux 64-bit systems.

To install the driver for an Amazon Linux AMI

1. Make sure the kernel-devel package is installed and matches the version of the kernel you are
    currently running.

    ```
    [root@ip-xxx ~]# yum  install kernel-devel- uname –r
    ```

2. Install the NVIDIA driver.

    ```
    [root@ip-xxx ~]# /root/NVIDIA-Linux-x86_64_260.19.12.run
    ```

3. Reboot the instance.
4. Confirm that the driver is functional.

    ```
    [root@ip-xxx ~]# /usr/bin/nvidia-smi -q -a
    ```

    The response lists the installed NVIDIA driver version and details about the GPUs.
5. Confirm that the GPUs are exposed.

    ```
    [root@ip-xxx ~]# ls /dev/*nv*
    /dev/nvidia0 /dev/nvidia1 /dev/nvidiactl /dev/nvram
    ```

If a newer version of the NVIDIA driver is available, you might want to update your instance to use it. To
update your instance, follow the steps in the preceding section for installing the driver and run the self-install
script from the newly downloaded driver version.

To install the CUDA toolkit package, download the package from the NVIDIA website and run the self-install
script as outlined in step 2 of the preceding section for installing the driver.

Cluster Instance Hardware Specifications

Hardware specifications for the cc1.4xlarge cluster compute instance type, cg1.4xlarge cluster GPU
instance type, and cc2.8xlarge cluster compute instance type are described in the following table.

Item	cc1.4xlarge cluster compute instance type or cg1.4xlarge cluster GPU instance type	cc2.8xlarge cluster compute instance type
Processor	33.5 EC2 Compute Units (2 x Intel Xeon X5570, quad-core "Nehalem" architecture)	88 EC2 Compute Units (2 x Intel Xeon E5-2670, eight-core "Sandy Bridge" architecture)
Hyperthreading	8 physical cores with 16 threads	16 physical cores with 32 threads
Memory	23 GiB Note: The cluster GPU instances can use up to 22 GiB, with 1 GiB reserved for GPU operation. The 22 GiB doesn't include the GPU on-board memory, which is 3 GiB per GPU for the NVIDIA Tesla M2050 GPU.	60.5 GiB
Platform	64-bit	64-bit
Network	10 Gbps Ethernet	10 Gbps Ethernet
Local Storage	1680 GiB instance 64-bit storage (2 x 840 GiB)	3360 GiB instance 64-bit storage (4 x 840 GiB)
GPUs (cluster GPU instances only)	Two NVIDIA Tesla M2050 "Fermi" GPUs Note: The instance must have the corresponding NVIDIA driver installed; the reference AMI we provide includes the driver. If you need to download and install the driver, see Manual Installation of the NVIDIA Driver (p. 108).	N/A

When determining the number of cluster instances you can launch, be aware that you're limited to a maximum of 20 EC2 instances, of which two can be cluster GPU instances. If you need more instances, go to the "Request to Increase Amazon EC2 Instance Limit" form at aws.amazon.com/contact-us/ec2-request.

Important

Please note the following:

- Sometimes you can get "Insufficient Capacity" errors when you try to add more instances to an existing placement group. If that is a concern, we recommend that you launch the minimum number of instances that you need in the cluster placement group at once in a single launch request. If you do receive an "Insufficient Capacity" error, stop and restart the instances in the placement group and attempt to add additional instances again.

- A cluster placement group cannot span multiple Availability Zones.

- A cluster placement group can only contain cluster instances of the same instance type (e.g, only cc2.8xlarge instances within a placement group).

Related Topics

- Instance Families and Types (p. 85)

Regions and Availability Zones

This section describes how to work with Regions and Availability Zones.

Topics

Region and Availability Zone Concepts

Amazon EC2 provides the ability to place instances in multiple locations. Amazon EC2 locations are composed of Availability Zones and Regions. Regions are dispersed and located in separate geographic areas (US, EU, etc.). Availability Zones are distinct locations within a Region that are engineered to be isolated from failures in other Availability Zones and provide inexpensive, low latency network connectivity to other Availability Zones in the same Region.

By launching instances in separate Regions, you can design your application to be closer to specific customers or to meet legal or other requirements. By launching instances in separate Availability Zones, you can protect your applications from the failure of a single location.

The following graphic shows a representation of Amazon EC2. Each Region is completely independent. Each Availability Zone is isolated, but connected through low-latency links.
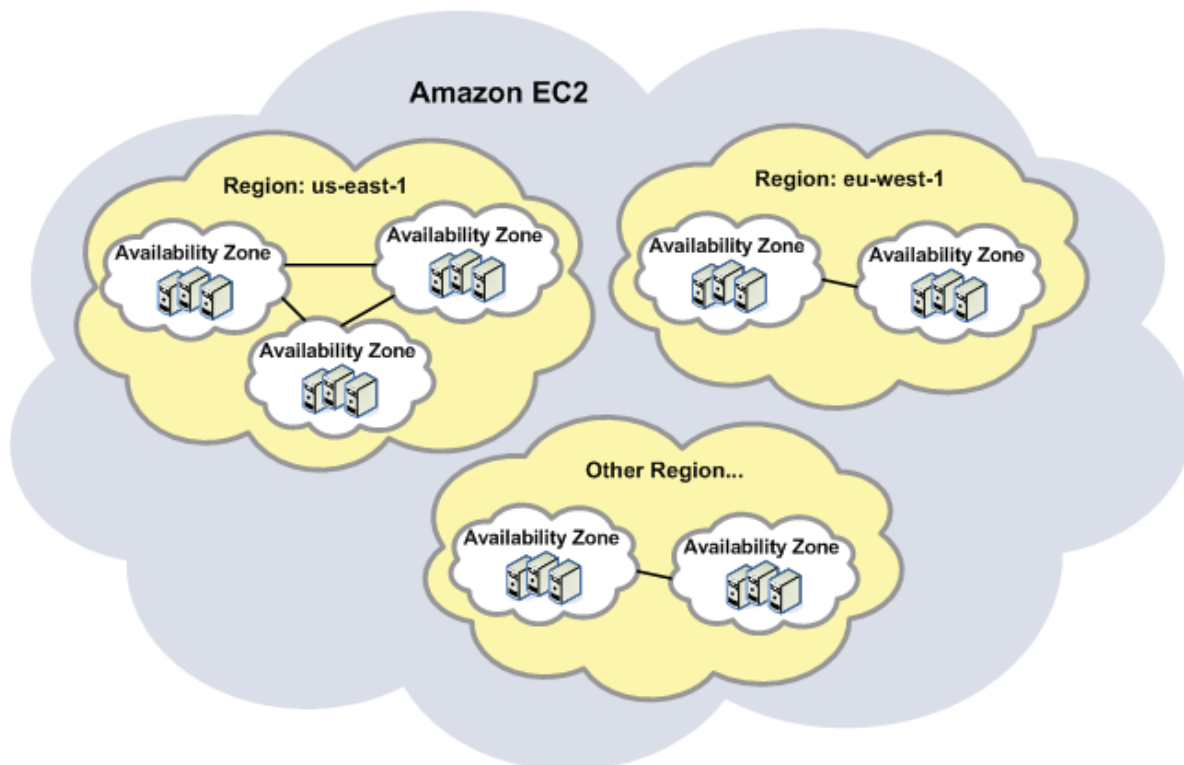
Regions

Amazon EC2 provides multiple Regions so you can launch Amazon EC2 instances in locations that meet your requirements. For example, you might want to launch instances in Europe to be closer to your European customers or to meet legal requirements.

Each Amazon EC2 Region is designed to be completely isolated from the other Amazon EC2 Regions. This achieves the greatest possible failure independence and stability, and it makes the locality of each EC2 resource unambiguous.

To launch or work with instances, you must specify the correct Region URL endpoint. For example, to access the US-East Region (default), you make service calls to the ec2.us-east-1.amazonaws.com service endpoint.

> **Note**
>
> Data transfer between Regions is charged at the Internet data transfer rate for both the sending and the receiving Region. For detailed information on Amazon EC2 charges, go to the Amazon EC2 Product Page.

For information about this product's regions and endpoints, go to Regions and Endpoints in the Amazon Web Services General Reference.

Availability Zones

Amazon operates state-of-the-art, highly available data center facilities. However, failures can occur that affect the availability of instances that are in the same location. Although this is rare, if you host all your Amazon EC2 instances in a single location that is affected by such a failure, your instances will be unavailable.

For example, if you have instances distributed across three Availability Zones and one of the instances fails, you can design your application so the instances in the remaining Availability Zones handle any requests.

> **Note**
>
> You can use Availability Zones in conjunction with elastic IP addresses to remap IP addresses across Availability Zones. For information on elastic IP addresses, see Elastic IP Address Concepts (p. 402).

API and Command Overview

The following table summarizes the available Region and Availability Zone commands and corresponding API actions. For more information about the commands, go to the Amazon Elastic Compute Cloud Command Line Reference. For more information about the API actions, go to the Amazon Elastic Compute Cloud API Reference.

Command and API Action	Description
ec2-describe-availability-zones DescribeAvailabilityZones	Lists Availability Zones available to your account
ec2-describe-regions DescribeRegions	Lists Regions available to your account

Describing Regions and Availability Zones

This section describes how to determine which Regions and Availability Zones are available.

AWS Management Console

To find Regions and Availability Zones

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. To determine available Regions, select the **Region** list box in the **Navigation** pane.

   After selecting a Region, you can view Availability Zones within that Region when launching an instance or creating a new Amazon EBS volume.
3. To view Availability Zones within a Region, click **Volumes** in the **Navigation** pane.

   The **Create Volume** dialog box appears.
4. To view Availability Zones within the Region, select the **Availability Zones** list box.
5. When you are finished, click **Cancel**.

Command Line Tools

To find Regions

- Enter the following command to describe Regions:

```
PROMPT> ec2-describe-regions
REGION   ap-northeast-1        ec2.ap-northeast-1.amazonaws.com
REGION   ap-southeast-1        ec2.ap-southeast-1.amazonaws.com
..
```

To find Availability Zones

- Enter the following command to describe Availability Zones within the `us-east-1` Region:

```
PROMPT> ec2-describe-availability-zones --region us-east-1
```

Amazon EC2 returns output similar to the following:

```
AVAILABILITYZONE           us-east-1a      available      us-east-1
AVAILABILITYZONE           us-east-1b      available      us-east-1
AVAILABILITYZONE           us-east-1c      available      us-east-1
AVAILABILITYZONE           us-east-1d      available      us-east-1
```

API

To find Regions

- Construct the following Query request.

```
https://ec2.amazonaws.com/
?Action=DescribeRegions
&AUTHPARAMS
```

The following is an example response.

```
<DescribeRegionsResponse xmlns="http://ec2.amazonaws.com/doc/2012-07-20/">
  <requestId>59dbff89-35bd-4eac-99ed-be587EXAMPLE</requestId>
  <regionInfo>
    <item>
      <regionName>us-east-1</regionName>
      <regionEndpoint>ec2.us-east-1.amazonaws.com</regionEndpoint>
    </item>
    <item>
      <regionName>eu-west-1</regionName>
      <regionEndpoint>ec2.eu-west-1.amazonaws.com</regionEndpoint>
    </item>
  </regionInfo>
</DescribeRegionsResponse>
```

To find Availability Zones

* Construct the following Query request.

```
https://ec2.amazonaws.com/
?Action=DescribeAvailabilityZones
&AUTHPARAMS
```

The following is an example response.

```
<DescribeAvailabilityZonesResponse xmlns="http://ec2.amazonaws.com/doc/2012-
07-20/">
  <requestId>35a9e5f3-4ed2-4cc2-a616-482b71c82c76</requestId>
    <availabilityZoneInfo>
      <item>
        <zoneName>us-east-1a</zoneName>
        <zoneState>available</zoneState>
        <regionName>us-east-1</regionName>
        <messageSet/>
      </item>
      <item>
        <zoneName>us-east-1b</zoneName>
        <zoneState>available</zoneState>
        <regionName>us-east-1</regionName>
        <messageSet/>
      </item>
      <item>
        <zoneName>us-east-1c</zoneName>
        <zoneState>available</zoneState>
        <regionName>us-east-1</regionName>
        <messageSet/>
      </item>
      <item>
```

```
            <zoneName>us-east-1d</zoneName>
            <zoneState>available</zoneState>
            <regionName>us-east-1</regionName>
            <messageSet/>
        </item>
    </availabilityZoneInfo>
</DescribeAvailabilityZonesResponse>
```

Specifying the Region to Use

Every command or call you make to EC2 must target a specific Region. This section explains how to specify the Region you want to use.

Note

The `ec2.us-west-1.amazonaws.com` Region is the original Amazon EC2 Region and is selected by default.

AWS Management Console

To specify the Region

* Use the Region selector in the top left corner of the page.
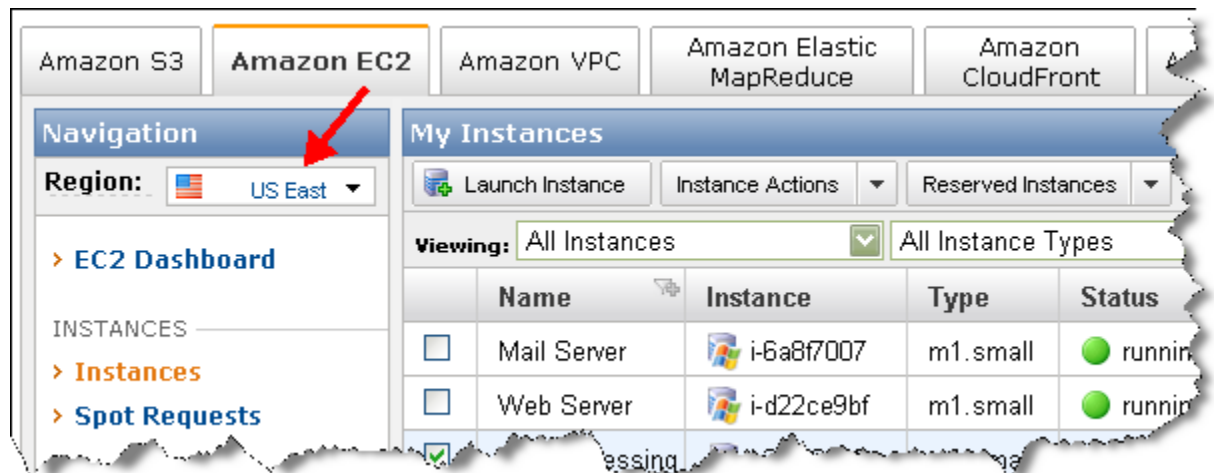
Command Line Tools

To specify the Region

* Change the `EC2_URL` environment variable's value to the Region's endpoint (e.g., https://ec2.us-west-1.amazonaws.com).

   Tip

   You can also use the `--region` command line option (e.g., `--region us-west-1`), or override the URL endpoint using the `-U` flag (e.g., `-U https://ec2.us-west-1.amazonaws.com`).

API

To specify the Region

- Configure your application to use the Region's endpoint (e.g., https://ec2.us-west-1.amazonaws.com).

Launching Instances in a Specific Availability Zone

When you launch an instance, you can optionally specify an Availability Zone. If you do not specify an Availability Zone, Amazon EC2 selects one for you in the Region that you are using. When launching your initial instances, we recommend accepting the default Availability Zone, which allows Amazon EC2 to select the best Availability Zone for you based on system health and available capacity. Even if you have other instances running, you might consider not specifying an Availability Zone if your new instances do not need to be close to, or separated from, your existing instances.

> **Note**
>
> Availability Zones are not the same across accounts. The Availability Zone us-east-1a for account A is not necessarily the same as us-east-1a for account B. Zone assignments are mapped independently for each account.
>
> You are charged a small bandwidth charge for data that crosses Availability Zones. For more information, go to the Amazon EC2 product page.

AWS Management Console

To launch an instance in a specific Availability Zone

1. Use the **Classic Wizard** to launch an instance.
2. On the **Instance Details** page of the instance launch wizard, you can specify the Availability Zone of your choice, or specify no preference.
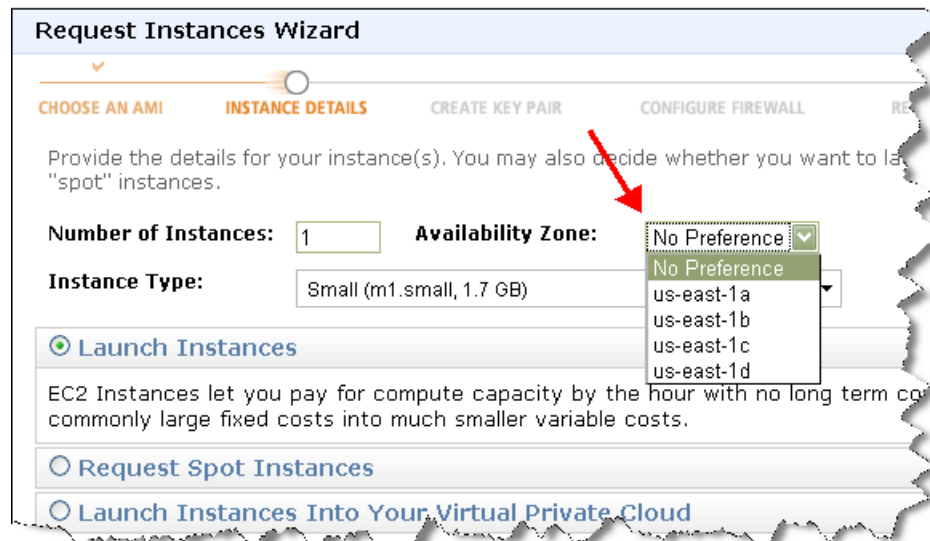
Command Line Tools

To launch an instance in a specific Availability Zone

- Specify the Availability Zone you want to launch the instances into using the `--availability-zone` option.

```
PROMPT> ec2-run-instances ami_id -n count --availability-zone zone
```

Amazon EC2 returns output similar to the following:

```
RESERVATION r-0ea54067 111122223333 default
INSTANCE i-cfd732a6 ami-6ba54002 pending 0 m1.small 2010-03-19T13:59:03+0000
 us-east-1a aki-94c527fd ari-96c527ff monitoring-disabled   ebs
INSTANCE i-cfd732a7 ami-6ba54002 pending 1 m1.small 2010-03-19T13:59:03+0000
 us-east-1a aki-94c527fd ari-96c527ff monitoring-disabled   ebs
INSTANCE i-cfd732a8 ami-6ba54002 pending 2 m1.small 2010-03-19T13:59:03+0000
 us-east-1a aki-94c527fd ari-96c527ff monitoring-disabled   ebs
INSTANCE i-cfd732a9 ami-6ba54002 pending 3 m1.small 2010-03-19T13:59:03+0000
 us-east-1a aki-94c527fd ari-96c527ff monitoring-disabled   ebs
```

API

To launch an instance in a specific Availability Zone

- Construct the following Query request.

```
https://ec2.amazonaws.com/
?Action=RunInstances
&ImageId=ami-id
&MaxCount=1
&MinCount=1
&KeyName=keypair-name
&Placement.AvailabilityZone=zone
&AUTHPARAMS
```

**Related Topics**

Root Device Volume

Topics

When you launch an Amazon EC2 instance, the root device volume contains the image used to boot the instance. When we introduced Amazon EC2, all AMIs were backed by Amazon EC2 instance store, which

means the root device for an instance launched from the AMI is an instance store volume created from a template stored in Amazon S3. After we introduced Amazon EBS, we introduced AMIs that are backed by Amazon EBS. This means that the root device for an instance launched from the AMI is an Amazon EBS volume created from an Amazon EBS snapshot. You can choose between Amazon EC2 instance store and Amazon EBS as the root device for your AMI. We recommend using AMIs backed by EBS, because they launch faster and use persistent storage.

Amazon EC2 Root Device Storage Concepts

An Amazon EC2 instance can be launched from one of two types of AMIs: an Amazon EC2 instance store-backed AMI or an Amazon EBS-backed AMI. The description of an AMI includes type of AMI it is; you'll see the root device referred to in some places as either `ebs` (for Amazon EBS-backed) or `instance store` (for Amazon EC2 instance store-backed). This is important because there are significant differences between what you can do with each type of AMI. For a discussion of these differences, see .

Instances that use instance store for the root device automatically have instance store volumes available, with a separate root partition. When an instance is launched, the image that is used to boot the instance is copied to the root partition. Any data on the instance store volumes persists as long as the instance is running and is deleted when the instance fails or terminates.
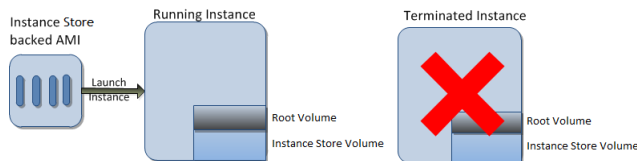
After an EC2 instance store-backed instance fails or terminates, it cannot be restored. If you plan to use Amazon EC2 instance store-backed instances, we highly recommend that you distribute the data on your instance stores across multiple Availability Zones. You should also back up the data on your instance store volumes to Amazon S3 on a regular basis.

Instances that use Amazon EBS for the root device automatically have an Amazon EBS volume attached. When an Amazon EBS-backed instance is launched, an EBS volume is created for each EBS snapshot referenced by the AMI. You must have at least one snapshot that denotes the root device; the others are optional and denote additional volumes to be created from other snapshots.

An Amazon EBS-backed instance can be stopped and later restarted without affecting data stored in the attached volumes. There are various instance- and volume-related tasks you can do when an Amazon EBS-backed instance is in a stopped state. For example, you can modify the properties of the instance, you can change the size of your instance or update the kernel it is using, or you can attach your root volume to a different running instance for debugging or any other purpose.
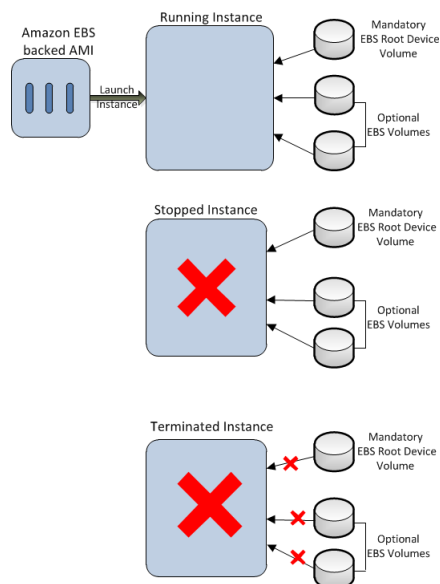
By default, the root device volume and the other volumes created when an Amazon EBS-backed instance is launched are automatically deleted when the instance terminates (in other words, their `DeleteOnTermination` flags are set to `true` by default). You can change the default behavior by setting the `DeleteOnTermination` flag to the value you want when you launch the instance. For an example of how to change the flag at launch time, see Using Amazon EC2 Root Device Storage (p. 122).

However, any EBS volumes that you attach *after* the instance is running (their `DeleteOnTermination` flags are set to `false` by default) are detached with their data intact on instance termination. These volumes can later be reattached to any running instance.

If an Amazon EBS-backed instance fails, you can restore your session by following one of these methods:

- Stop and then start again.
- Automatically snapshot all relevant volumes and create a new AMI. For more information, see Creating Amazon EBS-Backed AMIs (p. 36).
- Attach the volume to the new instance by following these steps:
  1. Snapshot just the root volume.
  2. Register a new AMI using the snapshot.
  3. Launch a new instance from the new AMI.
  4. Detach the remaining volumes from the old instance.
  5. Reattach the volumes to the new instance.

We recommend using either the first or the second method for failed instances with normal volume size and the third method for failed instances with large volumes.

Amazon EC2 Root Device Storage Usage Scenarios

You can implement Amazon EBS backed AMIs by creating a set of snapshots and registering an AMI that uses those snapshots. The AMI publisher controls the default size of the root device through the size of the snapshot. The default size can be increased up to 1TiB to accommodate the requirements of the application either at the time you register the EBS-backed AMI or while you launch the EBS-backed instance.

Launching an Amazon EBS-backed Instance with Increased Root Device Storage Disk Size

1	Select an Amazon EBS-backed AMI to launch your instance from.
2	Check the root device size and note the AMI ID.
3	Using the command line interface, launch the instance by specifying the AMI ID and the mapping of the root device with the increased size.
4	Connect to the instance.
5	Check the size of the root device on the instance. The increased size of the root device is not apparent yet. This is because the file system does not recognize the increased size on the root device.
6	Resize the file system.
7	Check the size of the root device.
8	The root device of the newly launched instance now shows the increased size.

> **Note**
>
> You cannot decrease the size of your root device to less than the size of the AMI. To decrease the size of your root device, create your own AMI with the desired size for the root device and then launch an instance from that AMI.

Increasing the Size of the Root Device on an Amazon EBS-Backed Running Instance

1	Get the Amazon EBS volume ID and the Availability Zone of a running instance for which you want to increase the root storage size.
2	Stop the instance.
3	Detach the original volume from the instance.
4	Create a snapshot of the detached volume.
5	Create a new volume from the snapshot by specifying a larger size.
6	Attach the new volume to the stopped instance.
7	Start the instance and get the new IP address/hostname.
8	Connect to the instance using the new IP address/hostname.
9	Resize the root file system to the extent of the new Amazon EBS volume.
10	Check the size of the root device. The root device nows shows the increased size.
11	[Optional] Delete the old EBS volume, if you no longer need it.

The following list describes the tasks for creating a snapshot of the root device of an Amazon EC2 instance store-backed instance. The snapshot is created using an Amazon EBS volume. You can use this snapshot to create a new EBS-backed AMI or to launch another instance.

Creating a Snapshot of the Root Device of an Amazon EC2 instance store-backed Instance

1	Launch an instance from an Amazon EC2 instance store-backed AMI.
2	Create a 10GiB Amazon EBS volume in the same Availability Zone as that of your newly launched instance. **Note** Use this volume to create a snapshot of the root partition of an Amazon EC2 instance store-backed AMI. The resulting snapshot is the same size as the root partition; the maximum size of the root partition in an Amazon EC2 instance store-backed AMI is 10GiB.
3	Attach the volume to the running instance using either the AWS Management Console or the command line tools.
4	Format the volume with a file system.
5	[For Linux Users] Create a directory and then mount the volume on the newly created directory.
6	Copy the data on the root storage device to the newly attached EBS volume.
7	Unmount and detach the volume from the instance.
8	Create a snapshot of the volume.

Amazon EC2 instance store-backed AMIs are limited to 10GiB storage for the root device. If you require additional storage on your root device, you must first convert the Amazon EC2 instance store-backed AMI to an Amazon EBS-backed AMI and then launch an EBS-backed instance with increased root storage.

> **Note**
>
> This conversion procedure works with a Linux AMI, but step 6 fails with a Windows AMI.

Converting an instance store-backed AMI to an EBS-backed AMI (Linux only)

1	Launch an instance from an Amazon EC2 instance store-backed AMI.
2	Create a 10GiB Amazon EBS volume in the same Availability Zone as that of your newly launched instance. Use this volume to create a snapshot of the root partition of the Amazon EC2 instance store-backed AMI. The resulting snapshot is the same size as the root partition; the maximum size of the root partition in an Amazon EC2 instance store-backed AMI is 10GiB.
3	Attach the volume to the running instance using either the AWS Management Console or the Command Line Tools.
4	Format the volume with a file system.
5	Create a directory and then mount the volume on the newly created directory.
6	Copy the data on the root storage device to the newly attached EBS volume.
7	Unmount and detach the volume from the instance.
8	Create a snapshot of the volume.

9	Register the snapshot of the volume as an AMI.

Using Amazon EC2 Root Device Storage

By default, the root device volume for an AMI backed by Amazon EBS is deleted when the instance terminates. This section describes how to change the default behavior of the root device volume.

Changing the Root Volume to Persist

In this section, we show you how to set the `DeleteOnTermination` flag to `false` in the instance's block device mapping at launch time.

Command Line Tools

To change the DeleteOnTermination flag at launch time

*   In your `ec2-run-instances` request, include a block device mapping that sets the `deleteOnTermination` flag for the root device to `false`. Include the `-v` option to run the command in verbose mode.

    ```
    PROMPT>  ec2-run-instances ami_id  -b root_device_name=::false other para
    meters... -v
    ```

    The root device is typically /dev/sda1, or xvda (for Windows). Following is an example.

    ```
    PROMPT>  ec2-run-instances ami-1a2b3c4d  -b /dev/sda1=::false other paramet
    ers... -v
    ```

    If you're using the command line tools on a Windows system, you must put quotation marks around the block device mapping value.

    ```
    PROMPT>  ec2-run-instances ami-1a2b3c4d  -b "xvda=::false" other parameters...
     -v
    ```

    By running the command in verbose mode, you can see the underlying SOAP request, and confirm that the `deleteOnTermination` value is set to `false`. The following XML snippet is from the SOAP request sent to Amazon EC2.

    ```
    ...
    <blockDeviceMapping>
        <item>
          <deviceName>/dev/sda1</deviceName>
          <ebs>
            <deleteOnTermination>false</deleteOnTermination>
          </ebs>
        </item>
    </blockDeviceMapping>
    ...
    ```

    You can also verify the setting after the instance launches by viewing the instance's details in the AWS Management Console.

API

To change the DeleteOnTermination flag at launch time

- Issue the following Query request to include a block device mapping that sets the
  `deleteOnTermination` flag for the root device to `false`. If it's a Windows AMI, use xvda instead
  of /dev/sda1 as the root device name.

```
https://ec2.amazonaws.com/
?Action=RunInstances
&ImageId=ami-1a2b3c4d
&BlockDeviceMapping.1.DeviceName=/dev/sda1
&BlockDeviceMapping.1.Ebs.DeleteOnTermination=false
&AUTHPARAMS
```

You can confirm the setting by using `DescribeInstances`. The following example snippet from the XML
response shows that the flag is set to `false` for the instance's root device.

```
...
 <rootDeviceName>/dev/sda1</rootDeviceName>
 <blockDeviceMapping>
   <item>
     <deviceName>/dev/sda1</deviceName>
     <ebs>
       <volumeId>vol-818843e8</volumeId>
       <status>attached</status>
       <attachTime>2010-02-22T20:36:18.000Z</attachTime>
       <deleteOnTermination>false</deleteOnTermination>
     </ebs>
   </item>
 </blockDeviceMapping>
...
```

You can also verify the setting by viewing the instance's details in the AWS Management Console. For
more information, see Viewing Block Device Mappings (p. 491).

Instance Best Practices

The instance is your basic computation building block. After you launch an instance, it 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 computer.

By default, you can run up to 20 instances. You can run as many of as few of these instances as you need at any time. If you need more than 20 instances, please complete the Amazon EC2 Instance Request Form.

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, or use Amazon EBS.

- 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 boot considerably faster.

- Monitor the health of your instances.
  For more information, go to the Amazon CloudWatch product page.

- Keep your Amazon EC2 firewall permissions as restrictive as possible.
  Only open up permissions that you require. Use separate *security 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. For more information about security groups, see Security Groups (p. 416) .

Spot Instances

If you have flexibility on when your application will run, you can bid on unused Amazon EC2 compute capacity, called Spot Instances, and lower your costs significantly. Set by Amazon EC2, the Spot Price for these instances fluctuates periodically depending on the supply of and demand for Spot Instance capacity.

To use Spot Instances, you place a Spot Instance request (your bid) specifying the maximum price you are willing to pay per hour per instance. If the maximum price of your bid is greater than the current Spot Price, your request is fulfilled and your instances run until you terminate them or the Spot Price increases above your maximum price. Your instance can also be terminated when your bid price equals the market price, even when there is no increase in the market price. This can happen when demand for capacity rises, or when supply fluctuates.

You will often pay less per hour than your maximum bid price. The Spot Price is adjusted periodically as requests come in and the available supply of instances changes. Everyone pays that same Spot Price for that period regardless of whether their maximum bid price was higher, and you will never pay more than your hourly maximum bid price.

Quick Look: Getting Started with Spot Instances Video

The following video shows how to launch your first Spot Instance using the AWS Management Console. This video includes instructions on placing a bid, determining when the instance is fulfilled, and canceling the instance. Getting Started with Spot Instances

Checklist for Getting Started with Spot Instances

If you want to get started working with Spot Instances, here are the resources you need to get going.

- Get Started with Spot Instances (p. 126)
  - View Spot Price History (p. 129)
  - Create a Spot Instance Request (p. 132)
  - Find Running Spot Instances (p. 137)
  - Cancel Spot Instance Requests (p. 141)
- Plan for Interruptions (p. 159)
- Bidding Strategies (p. 163)
- Advanced Tasks (p. 163)
  - Persist Your Root EBS Partition (p. 163)
  - Tag Spot Instance Requests (p. 164)
  - Launch Spot Instances in Amazon Virtual Private Cloud (p. 158)
 - Subscribe to Your Spot Instance Data Feed (p. 165)

What's New in Spot Instances

Here's a quick look at what's new in Spot Instances:

- Managing Spot Instances with Auto Scaling (p. 144)
- Get Notifications through Auto Scaling for Spot Instances (p. 160)

Get Started with Spot Instances

This section gives you a comprehensive overview of what you can do with Spot Instances. The following diagram gives you a high-level look at the tasks you can perform with Spot Instances.
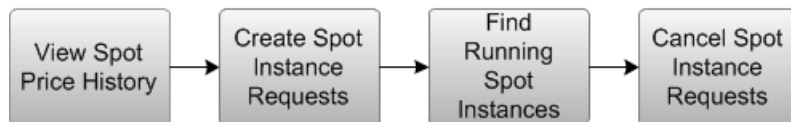
If you are new to Spot Instances, take a look at Prerequisites for Using Spot Instances (p. 126) to make sure you can take full advantage of the benefits of this Amazon EC2 product.

If you have been using Amazon EC2 and you're ready to look into making a Spot Instance request, proceed to one of the following steps:

- View Spot Price History (p. 129)
- Create a Spot Instance Request (p. 132)
- Find Running Spot Instances (p. 137)
- Cancel Spot Instance Requests (p. 141)
- Advanced Tasks (p. 163)

Prerequisites for Using Spot Instances

To work with Amazon EC2 Spot Instances, we assume you have read and completed the instructions described in the Getting Started with EC2, which provides information on creating your Amazon EC2 account and credentials.

If you use the Amazon EC2 command line interface (CLI) tools, we assume that you have read and completed the instructions described in the Getting Started with the Command Line Tools of the *Amazon EC2 User Guide*. It walks you through setting up your environment for use with the CLI tools.

In addition, whichever you choose to use—AWS Management Console, the CLI, or API—Amazon EC2 provides tools that specifically help you work with Spot Instances to assess Spot Price history, submit Spot Instance requests, and manage your Spot Instance requests (bids) and instances. You can also use programming languages to perform most of your work.

AWS Management Console

The AWS Management Console has tools specifically designed for Spot Instance request tasks. The console also has general tools that you can use to manage the instances launched when your Spot Instance requests are fulfilled. The following list identifies the tools you can use in the AWS Management Console:

- The **Spot Requests** page is the main way you interact with your Spot Instance requests.
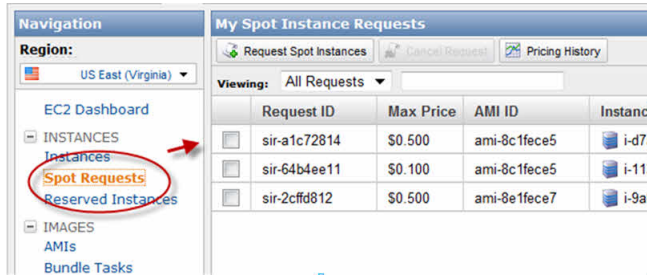
- **Spot Instance Pricing History** gives you an insight in the pricing patterns for specific Spot Instance types in Availability Zones over a defined period.
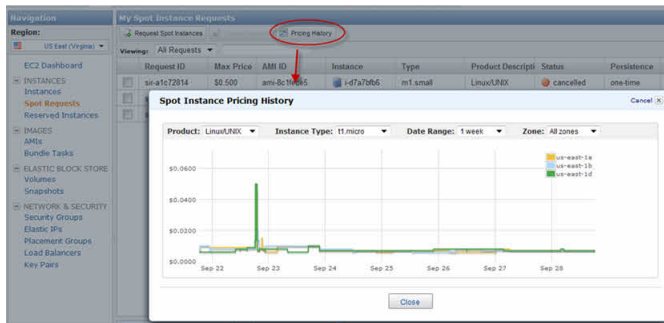
- Use the **Request Spot Instances** page to specify the details of instances to be launched when your Spot Instance bid succeeds.

- Use the **Instances** page to manage the instances launched when your Spot Instance bid succeeds.

Command Line Tools

To manage your Spot Requests, you use Amazon EC2 command line tools specifically designed for these tasks. To manage the instances launched when your Spot Request is fulfilled, use the same CLI commands that you use for Amazon EC2 instances.

The following table lists the CLI commands you use for Spot Request tasks.

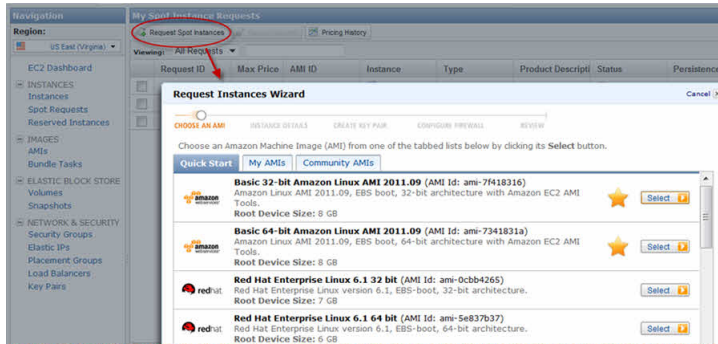| Task	CLI
View Spot Price History (p. 129).	`ec2-describe-spot-price-history`
Find Running Spot Instances (p. 137).	`ec2-describe-spot-instance-requests`
Create a Spot Instance Request (p. 132).	`ec2-request-spot-instances`
Subscribe to Your Spot Instance Data Feed (p. 166).	`ec2-create-spot-datafeed-subscription`
Data Feed Filename and Format (p. 165).	`ec2-describe-spot-datafeed-subscription`
Delete a Spot Instance Data Feed (p. 167).	`ec2-delete-spot-datafeed-subscription`
Cancel Spot Instance Requests (p. 141).	`ec2-cancel-spot-instance-requests`

For information about CLI commands, go to the Amazon Elastic Compute Cloud Command Line Reference.

API

To manage your Spot Requests, you use Amazon EC2 API tools specifically designed for these tasks. To manage the instances launched when your Spot Request is fulfilled, use the same API actions that you use for Amazon EC2 instances.

The following table lists the API actions you use for Spot Request tasks.

Task	API
View Spot Price History (p. 129).	`DescribeSpotPriceHistory`
Find Running Spot Instances (p. 137).	`DescribeSpotInstanceRequests`
Create a Spot Instance Request (p. 132).	`RequestSpotInstances`
Subscribe to Your Spot Instance Data Feed (p. 166).	`CreateSpotDatafeedSubscription`
Data Feed Filename and Format (p. 165).	`DescribeSpotDatafeedSubscription`
Delete a Spot Instance Data Feed (p. 167).	`DeleteSpotDatafeedSubscription`
Cancel Spot Instance Requests (p. 141).	`CancelSpotInstanceRequests`

For information about API actions, go to the Amazon Elastic Compute Cloud API Reference.

AWS Java SDK

Java developers can go to the *AWS SDK for Java* to consult the growing library of tutorials on Spot Instances:

- Tutorial for Java Developers: Amazon EC2 Spot Instances
- Tutorial for Java Developers: Advanced Amazon EC2 Spot Instance Request Management

View Spot Price History

Before specifying a rate you want to bid for your Spot Instance, we recommend that you view the Spot Price history. You can view the Spot Price history over a period from one to 90 days based on the instance type, the operating system you want the instance to run on, the time period, and the Availability Zone in which it will be launched.

For example, let's say you want to bid for a Linux/UNIX micro instance to be launched in the us-east-1a Availability Zone. Specify these values in the API or select them in the **Spot Price History** page of the AWS Management Console. You will be able to view the price for the instance type and operating system in the Availability Zone you want during the period you specified. On the other hand, if you don't need to launch the instances in a specific Availability Zone and consequently you don't specify this option, Amazon EC2 returns the prices for all Availability Zones.

Keep in mind that if you use the `DescribeSpotPriceHistory` action or `ec2-describe-spot-price-history` command before the 2011-05-15 API version, you will get the lowest price across the Region for the given time period and the prices will be returned in chronological order.

Note

Make sure you have set up the prerequisites to working with Amazon EC2. If you haven't, go to Prerequisites for Using Spot Instances (p. 126).

AWS Management Console

To view Spot Price history

1. From the Amazon EC2 console, click **Spot Requests** in the navigation pane.
   The **My Spot Instance Requests** pane opens.

2. At the top of the **My Spot Instance Requests** pane, click the **Pricing History** button.
   The console displays the Spot Instance pricing history.

   Note

   Using the historical pricing as a guide, select a price that you think would be likely to keep your instances running for the period of time you need.

3. If you want to view the Spot Price history for specific Availability Zones, click the **Zone** drop-down list and select an Availability Zone.
   The **Spot Instance Pricing History** page displays the Spot Instance pricing history for the zone you selected.
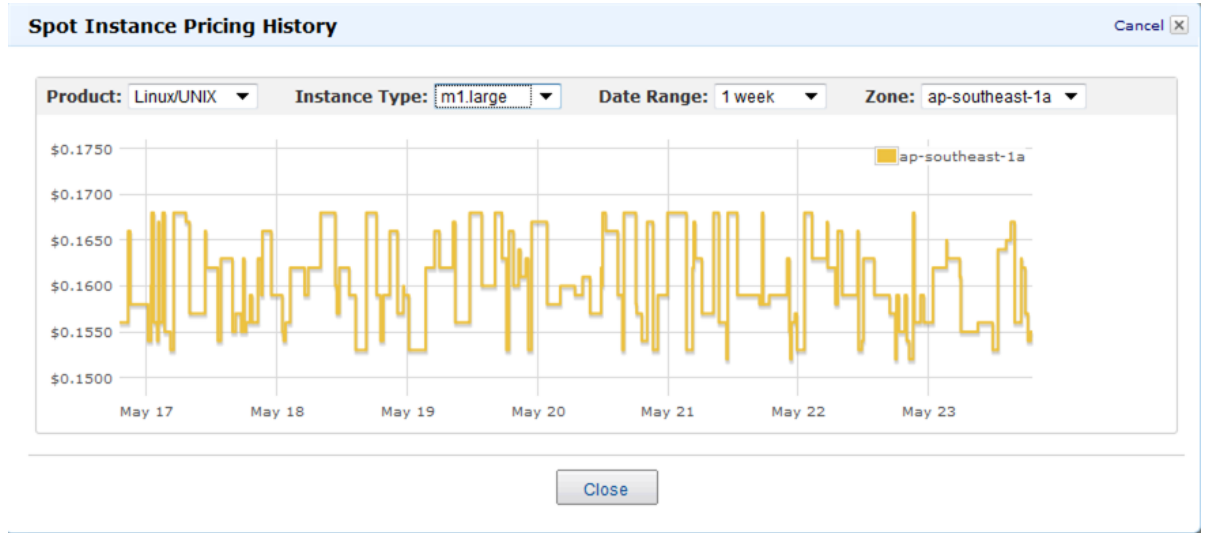
Command Line Tools

To view Spot Price history

1. Enter the following command:

```
PROMPT> ec2-describe-spot-price-history -H --instance-type m1.xlarge
```

Amazon EC2 returns output similar to the following:

```
SPOTINSTANCEPRICE    0.384000    2011-05-25T11:37:48-0800    m1.xlarge    Windows
 us-east-1b
SPOTINSTANCEPRICE    0.384000    2011-05-25T11:37:48-0800    m1.xlarge    Windows
 us-east-1d
…
SPOTINSTANCEPRICE    0.242000    2011-04-18T14:39:14-0800    m1.xlarge    SUSE
 Linux   us-east-1d
SPOTINSTANCEPRICE    0.242000    2011-04-18T14:39:14-0800    m1.xlarge    SUSE
 Linux   us-east-1a
```

In this example, the price for the `m1.xlarge` instance type ranges between $0.242 and $0.384.

2. Based on the historical pricing, select a price that is likely to keep your instances running for the period of time that you need.

Tip

You can filter the spot history data so it includes only instance types or dates of interest to you. For more information about how to filter the results, go to ec2-describe-spot-price-history in the *Amazon Elastic Compute Cloud Command Line Reference*.

API

To view Spot Price history

- Construct the following Query request.

```
https://ec2.amazonaws.com/
?Action=DescribeSpotPriceHistory
&InstanceType=instance_type
&...auth parameters...
```

Following is an example response.

```
<DescribeSpotPriceHistoryResponse xmlns="http://ec2.amazonaws.com/doc/2012-
07-20/">
    <spotPriceHistorySet>
      <item>
        <instanceType>m1.small</instanceType>
        <productDescription>Linux/UNIX</productDescription>
        <spotPrice>.28</spotPrice>
        <timestamp>2009-12-01T11:51:50.000Z</timestamp>
        <availabilityZone>us-east-1a</availabilityZone>
      </item>
      <item>
        <instanceType>m1.small</instanceType>
        <productDescription>Linux/UNIX</productDescription>
        <spotPrice>.28</spotPrice>
        <timestamp>2009-12-01T11:51:50.000Z</timestamp>
        <availabilityZone>us-east-1a</availabilityZone>
      </item>
      <item>
        <instanceType>m1.small</instanceType>
        <productDescription>Linux/UNIX</productDescription>
        <spotPrice>.31</spotPrice>
        <timestamp>2009-12-01T11:51:50.000Z</timestamp>
        <availabilityZone>us-east-1b</availabilityZone>
      </item>
      <item>
        <instanceType>m1.small</instanceType>
        <productDescription>Linux/UNIX</productDescription>
        <spotPrice>.30</spotPrice>
        <timestamp>2009-12-01T11:51:50.000Z</timestamp>
        <availabilityZone>us-east-1b</availabilityZone>
      </item>
      <item>
        <instanceType>m1.small</instanceType>
        <productDescription>Linux/UNIX</productDescription>
        <spotPrice>.25</spotPrice>
        <timestamp>2009-12-01T11:51:50.000Z</timestamp>
        <availabilityZone>us-east-1c</availabilityZone>
      </item>
      <item>
        <instanceType>m1.small</instanceType>
        <productDescription>Linux/UNIX</productDescription>
        <spotPrice>.28</spotPrice>
        <timestamp>2009-12-01T11:51:50.000Z</timestamp>
        <availabilityZone>us-east-1c</availabilityZone>
```

```
      </item>
      <item>
        <instanceType>m1.small</instanceType>
        <productDescription>Linux/UNIX</productDescription>
        <spotPrice>.35</spotPrice>
        <timestamp>2009-12-01T11:51:50.000Z</timestamp>
        <availabilityZone>us-east-1c</availabilityZone>
      </item>
   </spotPriceHistorySet>
   <nextToken/>
</DescribeSpotPriceHistoryResponse>
```

Note

Based on the historical pricing, select a price that is likely to keep your instances running
for the period of time that you need.

Tip

You can filter the spot history data so it includes only instance types or dates of interest to you.
For more information about how to filter the results, go to DescribeSpotPriceHistory in the *Amazon
Elastic Compute Cloud API Reference*.

What do you want to do next?

- Create a Spot Instance Request (p. 132)
- Find Running Spot Instances (p. 137)
- Bidding Strategies (p. 163)
- Launch Spot Instances in Amazon Virtual Private Cloud (p. 158)

Create a Spot Instance Request

After deciding on a maximum price, you are ready to request Spot Instances. The maximum price is not
necessarily the price that you will pay. For example, if you specify $.05 as your maximum price and the
Spot Price is $.03 for the period, you will pay $.03. If the Spot Price drops, you will pay less. If the Spot
Price increases, you will pay the new price (up to your maximum).

If your maximum price is greater than the Spot Price and all additional constraints are met, Amazon EC2
automatically launches instances on your behalf.

If your Spot Instances request is for multiple instances, Amazon EC2 creates a separate Spot Instance
request ID (e.g., sir-1a2b3c4d) for each instance. You can then track the status of each of the requests
separately.

This section discusses the details of creating requests for Spot Instances to be launched in Amazon EC2.
You can also launch Spot Instances in the Amazon Virtual Private Cloud (Amazon VPC) and the steps
are similar, but they will not be discussed here. For more information about launching Spot Instances in
Amazon VPC, see Launch Spot Instances in Amazon Virtual Private Cloud (p. 158).

Before requesting a Spot Instance, ensure you have an Amazon Machine Image (AMI) that does the
following for your instance:

- Automatically performs the tasks you want at start-up because the instance will start asynchronously
  without notification.

- Stores important data regularly and in a place that won't be affected by instance termination, such as Amazon S3, Amazon SimpleDB, or Amazon EBS.

- Handles termination gracefully.

For information about creating AMIs, see Creating Your Own AMIs (p. 35).

> **Important**
>
> Although Spot Instances can use Amazon EBS-backed AMIs, be aware that the EBS-backed AMIs don't support Stop/Start. In other words, you can't stop and start Spot Instances launched from an AMI with an Amazon EBS root device.

Types of Spot Instance Requests

You can make two types of Spot Instance requests—a one-time request or a persistent request. A one-time request remains active until either all the requested instances launch, the request expires, or you cancel the request. For example, if you create a one-time request for three instances, the request is considered complete after all three instances launch.

Persistent Spot Instance requests remain active until they expire or you cancel them, even if the request was previously satisfied. For example, if you create a persistent Spot Instance request for one instance at $.30, Amazon EC2 launches or keeps your instance running if your maximum bid price is above $.30 and terminates your instance if your maximum bid price is below $.30.

With both one-time and persistent requests, instances continue to run until either they no longer exceed the Spot Price, you terminate them, or they terminate on their own. If the maximum price is exactly equal to the Spot Price, an instance might or might not continue running (depending on available capacity).

Launching Spot Instances in Launch Groups and Availability Zones

You can opt to have your Spot Instances launch at the same time or in the same Availability Zone. To tell Amazon EC2 to launch your instances only if all the instances in the request can be fulfilled, specify a Launch Group in your Spot Instance request.

To specify a launch group in the AWS Management Console, you select **Launch Group** in the Request Instances Wizard; with the CLI or API tools, you specify the `launch group` option when you make the request spot instances call.

If you want all your instances to be launched together in a single Availability Zone, specify an Availability Zone Group. Do this by selecting **Availability Zone Group** when using the Request Instances Wizard in the AWS Managment Console, or by specifying the `availability-zone-group` option in the CLI or API.

Although the Launch Group and Availability Zone Group specifications can be advantageous, use them only when needed. Avoiding these requirements increases the chances that your Spot Instance request can be fulfilled.

When you want to identify a bid price for a Spot Instance in a specific Availability Zone, just view the price history for the Availability Zone where you want to make your bid. Do this by selecting the Availability Zone from the **Zone** drop-down menu in the **Spot Instance Pricing History** page in the AWS Management Console. Alternatively, call `DescribeSpotPriceHistory` or use the `ec2-describe-spot-price-history` CLI command. You will see the price history for the specified Availability Zone with the most recent set of prices listed first. If you don't specify an Availability Zone, you will get the lowest prices across all Availability Zones for the time period, starting with the most recent set.

Note

When you use the `DescribeSpotPriceHistory` action or the
`ec2-describe-spot-price-history` command before the 2011-05-15 API version, you will
get the lowest price across the Region for the given time period and the prices will be returned
in chronological order—that is, from the oldest to the most recent.

For more information about Availability Zones, see Region and Availability Zone Concepts (p. 111).

Note

Make sure you have set up the prerequisites for working with Amazon EC2. If you haven't, go
to Prerequisites for Using Spot Instances (p. 126).

AWS Management Console

To create a Spot Price request

1. From the Amazon EC2 console, click **Spot Requests** in the navigation pane.
    The **My Spot Instance Request** pane opens.

2. Click the **Request Spot Instances** button in the right pane.
    The Request Instances Wizard starts.

3. Choose an AMI and click **Select**.
    The **INSTANCE DETAILS** page appears with the **Request Spot Instances** button already selected.

4. Specify the number and type of instances you want, and the desired Availability Zone (if any).

    Note

    Current Price displays the price for the Availability Zone you specified. If you do not specify
    an Availability Zone, **Current Price** displays the lowest price across all Availability Zones.
    The information displayed when the cursor hovers over the **Current Price** shows the price
    for each of the Availability Zones.

5. Configure the Spot Instance settings using the following table as a guide, and then click **Continue**.

Option	Description
Max Price	Specifies the maximum price you are willing to pay per instance hour.
Request Valid From and **Request Valid Until**	Defines the validity period of a Spot Request. The valid period, beginning with **Request Valid From** through **Request Valid Until**, specifies the length of time that your request will remain valid. By default, a Spot Request will be considered for fulfillment from the time it is created until it is either fulfilled or canceled by you. However, you can constrain the validity period using these options. **Note** The end time you specify doesn't apply to the Spot Instances launched by this request. This end time only applies to the Spot Instance request.
Persistent Request?	Determines whether your request is one-time or persistent. By default, it is one-time. A one-time request can only be fulfilled once. A persistent request is considered for fulfillment whenever there is no Spot Instance running for the same request.
Launch Group	Groups a set of requests together in the same launch group. All requests in a launch group have their instances started and terminated together.
Availability Zone Group	Groups a set of requests together in the same Availability Zone. All requests that share an Availability Zone group will launch Spot Instances in the same Availability Zone.

6.   Continue with the launch wizard as you normally would when launching instances.

If your request specified multiple instances, EC2 creates a separate Spot Instance request for each instance. The requests are displayed on the **Spot Requests** page.

Command Line Tools

To create a Spot Price request

• Enter the following command:

```
PROMPT> ec2-request-spot-instances
--price price
--user-data data
--instance-count count
--type one-time | persistent
[--valid-from timestamp]
```

```
[--valid-until timestamp]
[--launch-group launchgroup]
(run-instances-arguments)
```

For example:

```
PROMPT> ec2-request-spot-instances --price 0.32 ami-1234abcd --key MyKeypair
 --group websrv --instance-type m1.small --instance-count 10 --type one-time
```

Amazon EC2 returns output similar to the following:

```
SPOTINSTANCEREQUEST     sir-09fb0a04     0.32     one-time          Linux/UNIX
    open     2010-04-06T10:03:09+0200     ami-1234abc     m1.small
MyKeypair     websrv     monitoring-disabled
...
```

If your request specified multiple instances, EC2 creates a separate Spot Instance request for each instance; each instance has a different ID (e.g., sir-09fb0a04).

API

To create a Spot Price request

- Construct a Query request similar to the following.

```
https://ec2.amazonaws.com/
?Action=RequestSpotInstances
&SpotPrice.1=0.50
&InstanceCount.1=10
&Type.1=one-time
&AvailabilityZoneGroup.1=MyAzGroup
&LaunchSpecification.ImageId.1=ami-43a4412a
&LaunchSpecification.KeyName.1=MyKeypair
&LaunchSpecification.GroupSet.1=websrv
&LaunchSpecification.InstanceType.1=m1.small
&...auth parameters...
```

Following is an example response.

```
<RequestSpotInstancesResponse xmlns="http://ec2.amazonaws.com/doc/2012-07-
20/">
    <spotInstanceRequestSet>
        <item>
            <spotInstanceRequestId>sir-876aff12</spotInstanceRequestId>
            <spotPrice>0.32</spotPrice>
            <type>one-time</type>
            <state>open</state>
            <fault/>
            <validFrom/>
            <validUntil/>
            <launchGroup/>
            <availabilityZoneGroup>MyAzGroup</availabilityZoneGroup>
```

```
            <launchSpecification>
                <imageId>ami-43a4412a</imageId>
                <keyName>MyKeypair</keyName>
                <groupSet>
                    <item>
                        <groupId>websrv</groupId>
                    </item>
                </groupSet>
                <instanceType>m1.small</instanceType>
                ...
            </launchSpecification>
            <instanceId/>
            <createTime>2009-10-19T00:00:00+0000</createTime>
            <productDescription>Linux/UNIX</productDescription>
            <tagSet/>
        </item>
        ...
    </spotInstanceRequestSet>
</RequestSpotInstancesResponse>
```

If your request specified multiple instances, EC2 creates a separate Spot Instance request for each instance; each instance has a different ID (e.g., sir-876aff12).

What do you want to do next?

Find Running Spot Instances

Spot Instances continue running until the Spot Price is greater than your maximum bid price. In addition, when your Spot Instance request is canceled, the Spot Instances that were launched previously through the canceled request don't automatically get terminated. The only time that the Spot Instance service terminates a running instance is when the Spot Price exceeds your price. This section describes how to find running Spot Instances.

Important

Although Spot Instances can use Amazon EBS-backed AMIs, be aware that the EBS-backed AMIs don't support Stop/Start. In other words, you can't stop and start Spot Instances launched from an AMI with an Amazon EBS root device.

Note

Make sure you have set up the prerequisites for working with Amazon EC2. If you haven't, go to Prerequisites for Using Spot Instances (p. 126).

AWS Management Console

To find running Spot Instances

1. From the Amazon EC2 console, click **Instances** in the **Navigation** pane.
   The console displays a list of running instances.

2. Look for instances in the table where the **Lifecycle** column contains *spot*. You might need to turn on the display of the column (click **Show/Hide** in the top right corner of the page).

Command Line Tools

To find running Spot Instances

- Use `ec2-describe-spot-instance-requests`. If your Spot Instance request has been fulfilled (an instance has been launched), the instance ID appears in the response.

```
PROMPT> ec2-describe-spot-instance-requests
SPOTINSTANCEREQUEST     sir-e1471206    0.09     one-time        Linux/UNIX
    active   2010-09-13T16:50:44-0800
i-992cf7dd        ami-813968c4    m1.small       MyKey    default
            monitoring-disabled
```

- You can alternately use `ec2-describe-instances` with the following filter: `--filter instance-lifecycle=spot`. If you're using the command line tools on a Windows system, you might need to use quotation marks (`--filter "instance-lifecycle=spot"`). For more information about filters, see Listing and Filtering Your Resources (p. 503).

```
PROMPT>  ec2-describe-instances --filter instance-lifecycle=spot
```

EC2 returns output similar to the following:

```
RESERVATION     r-b58651f1      111122223333    default
INSTANCE        i-992cf7dd      ami-813968c4    ec2-184-72-8-111.us-west-
1.compute.amazonaws.com   ip-10-166-105-139.us-west-1.compute.internal
  running    MyKey   0              m1.small        2010-09-13T23:54:40+0000
      us-west-1a      aki-a13667e4    ari-a33667e6
  monitoring-disabled  184.72.8.111    10.166.105.139                  ebs
    spot    sir-e1471206            paravirtual
```

API

To find running Spot Instances

1. Construct a `DescribeInstances` Query request and use a filter to look for instances where `instance-lifecycle=spot`. For more information about filters, see Listing and Filtering Your Resources (p. 503).

```
https://ec2.amazonaws.com/
?Action=DescribeInstances
&Filter.1.Name=instance-lifecycle
&Filter.1.Value.1=spot
&...auth parameters...
```

Following is an example response. It includes an `instanceLifecycle` element with `spot` as the value.

```
<DescribeInstancesResponse xmlns="http://ec2.amazonaws.com/doc/2012-07-20/">

  ...
   <instancesSet>
   <item>
      <instanceId>i-992cf7dd</instanceId>
      <imageId>ami-813968c4</imageId>
      <instanceState>
         <code>16</code>
         <name>running</name>
      </instanceState>
    <privateDnsName>ip-10-166-105-139.us-west-1.compute.internal</privateDns
Name>
      <dnsName>ec2-184-72-8-111.us-west-1.compute.amazonaws.com</dnsName>
      <reason/>
      <keyName>MyKey</keyName>
      <amiLaunchIndex>0</amiLaunchIndex>
      <productCodes/>
      <instanceType>m1.small</instanceType>
      <launchTime>2010-09-13T23:54:40.000Z</launchTime>
      <placement>
         <availabilityZone>us-west-1a</availabilityZone>
         <groupName/>
      </placement>
      <kernelId>aki-a13667e4</kernelId>
      <ramdiskId>ari-a33667e6</ramdiskId>
      <monitoring>
         <state>disabled</state>
      </monitoring>
      <privateIpAddress>10.166.105.139</privateIpAddress>
      <ipAddress>184.72.8.111</ipAddress>
      <architecture>i386</architecture>
      <rootDeviceType>ebs</rootDeviceType>
      <rootDeviceName>/dev/sda1</rootDeviceName>
      <blockDeviceMapping>
         <item>
            <deviceName>/dev/sda1</deviceName>
            <ebs>
               <volumeId>vol-61088f0a</volumeId>
               <status>attached</status>
```

```
            <attachTime>2010-09-13T23:54:42.000Z</attachTime>
            <deleteOnTermination>true</deleteOnTermination>
          </ebs>
        </item>
      </blockDeviceMapping>
      <instanceLifecycle>spot</instanceLifecycle>
      <spotInstanceRequestId>sir-e1471206</spotInstanceRequestId>
      <virtualizationType>paravirtual</virtualizationType>
    </item>
  </instancesSet>
</DescribeInstancesResponse>
```

2. You can alternately use `DescribeSpotInstanceRequests`. If your Spot Instance request has been fulfilled (an instance has been launched), the instance ID appears in the response. Following is an excerpt from a response.

```
...
<spotInstanceRequestSet>
    <item>
        <spotInstanceRequestId>sir-e1471206</spotInstanceRequestId>
        <spotPrice>0.09</spotPrice>
        <type>one-time</type>
        <state>active</state>
        <launchSpecification>
            <imageId>ami-813968c4</imageId>
            <keyName>MyKey</keyName>
            <groupSet>
                <item>
                    <groupId>default</groupId>
                </item>
            </groupSet>
            <instanceType>m1.small</instanceType>
            <blockDeviceMapping/>
            <monitoring>
                <enabled>false</enabled>
            </monitoring>
        </launchSpecification>
        <instanceId>i-992cf7dd</instanceId>
        <createTime>2010-09-13T23:50:44.000Z</createTime>
        <productDescription>Linux/UNIX</productDescription>
        <launchedAvailabilityZone>us-east-1c</launchedAvailabilityZone>
    </item>
<spotInstanceRequestSet/>
...
```

What do you want to do next?

- Subscribe to Your Spot Instance Data Feed (p. 165)

Cancel Spot Instance Requests

Each Spot Instance request can be in one of the following states:

- **Open—**The request is not fulfilled.
- **Active—**The request is currently active (fulfilled).
- **Closed—**The request either completed or was not fulfilled within the period specified.

  Depending on conditions, an instance might still be running.

- **Cancelled—**The request is canceled because one of two events took place: You canceled the request, or the bid request went past its expiration date.

  In addition, when your Spot Instance request is canceled, either because the bid request went beyond its expiration date or you manually canceled it, the Spot Instances that were launched previously through the now-canceled request don't automatically get terminated. The only time that the Spot Instance service terminates a running instance is when the Spot Price equals or exceeds your price. Also, you can always terminate your instances manually.

  Note

  The **Valid Until** option that you specify when you submit a Spot Instance request applies only to the *request*; this deadline doesn't apply to Spot Instances that are *launched* by the request.

After placing a request, you can check its state and the state of your other requests.

Note
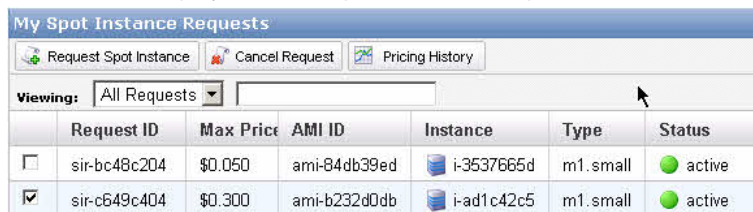
Make sure you have set up the prerequisites for working with Amazon EC2. If you haven't, go to Prerequisites for Using Spot Instances (p. 126).

AWS Management Console

To cancel Spot Instance requests

1. From the Amazon EC2 console, click **Spot Requests** in the navigation pane.

   The console displays a list of Spot Instances requests.

2. Select the spot instances you want to cancel and click the **Cancel Request** button.

Command Line Tools

To cancel Spot Instance requests

1. Enter the following command:

```
PROMPT> ec2-describe-spot-instance-requests
```

Amazon EC2 returns output similar to the following:

```
SPOTINSTANCEREQUEST    sir-09fb0a04    0.04    one-time    Linux/UNIX    closed

2010-04-06T10:03:09+0200    ami-b232d0db    m1.small    gsg-keypair    default
  monitoring-disabled
```

2. To cancel the request, enter the following:

```
PROMPT> ec2-cancel-spot-instance-requests sir-09fb0a04
```

Amazon EC2 returns output similar to the following:

```
SPOTINSTANCEREQUEST        sir-09fb0a04      cancelled
```

Tip

You can filter the list of Spot Instance requests to return only certain types of interest to you. For
more information about how to filter the results, go to ec2-describe-spot-instance-requests in
the *Amazon Elastic Compute Cloud Command Line Reference*.

API

To cancel Spot Instance requests

1. Construct the following Query request.

```
https://ec2.amazonaws.com/
?Action=DescribeSpotInstanceRequests
&...auth parameters...
```

Following is an example response.

```
<DescribeSpotInstanceRequestsResponse xmlns="http://ec2.amazon
aws.com/doc/2012-07-20/">
   <spotInstanceRequestSet>
      <item>
         <spotInstanceRequestId>sir-8675309a</spotInstanceRequestId>
         <spotPrice>0.32</spotPrice>
         <type>one-time</type>
         <state>open</state>
         <fault/>
         <validFrom/>
         <validUntil/>
         <launchGroup/>
         <availabilityZoneGroup>MyAzGroup</availabilityZoneGroup>
         <launchSpecification>
             <imageId> i-43a4412a</imageId>
             <keyName>MyKeypair</keyName>
```

```
            <groupSet>websrv</groupSet>
            <instanceType>m1.small</instanceType>
        </launchSpecification>
        <instanceId>i-123345678</instanceId>
        <createTime>2009-10-19T00:00:00+0000</createTime>
      <productDescription>Linux/UNIX</productDescription>
      <launchedAvailabilityZone>us-east-1c</launchedAvailabilityZone>
      </item>
   </spotInstanceRequestSet>
</DescribeSpotInstanceRequestsResponse>
```

2. Construct a Query request to cancel the Spot Instance requests.

```
https://ec2.amazonaws.com/
?Action=CancelSpotInstanceRequests
&SpotInstanceRequestId.1=sir-8675309a
&...auth parameters...
```

Following is an example response.

```
<CancelSpotInstanceRequestsResponse xmlns="http://ec2.amazonaws.com/doc/2012-
07-20/">
   <spotInstanceRequestId>sir-8675309a</spotInstanceRequestId>
</CancelSpotInstanceRequestsResponse>
```

Tip

You can filter the list of Spot Instance requests to return only certain types of interest to you. For
more information about how to filter the results, go to DescribeSpotInstanceRequests in the
Amazon Elastic Compute Cloud API Reference.

What do you want to do next?

- Create a Spot Instance Request (p. 132)
- View Spot Price History (p. 129)
- Find Running Spot Instances (p. 137)
- Launch Spot Instances in Amazon Virtual Private Cloud (p. 158)

Walkthroughs: Using Spot Instances with AWS Services

Topics
- Managing Spot Instances with Auto Scaling (p. 144)
- Using CloudFormation Templates to Launch Spot Instances (p. 156)
- Launching Amazon Elastic MapReduce Job Flows with Spot Instances (p. 157)
- Launch Spot Instances in Amazon Virtual Private Cloud (p. 158)

You can use AWS services with Spot Instances. In this section, we will show you how Amazon EC2 Spot Instances works with services, such as Auto Scaling, Elastic MapReduce, and Amazon Virtual Private Cloud (Amazon VPC).

Managing Spot Instances with Auto Scaling

Topics

You can take advantage of Auto Scaling features to manage your Amazon EC2 Spot Instances. With Auto Scaling, you can scale up or down your capacity based on demand by setting up Auto Scaling to make Spot bids on your behalf.

In addition, you can use Auto Scaling's scheduling functionality for more granular control over when to bid and launch your Spot Instances. You also can use an Amazon Simple Notification Service (Amazon SNS)-backed Auto Scaling feature that sends notifications each time specified events—such as the launch or termination of instances—take place. Using Auto Scaling's scheduled actions, you can set up bids to expire at a set time.

When you use Auto Scaling with Spot Instances, there are some Auto Scaling tools that you have to use instead of the Spot Instance tools that might be familiar. There are also a number of Spot Instance options that you cannot use. Here's a summary of differences:

- **Setting your bid price.** When you use Auto Scaling to launch Spot Instances, you set your bid price in an Auto Scaling launch configuration.
- **Spot market price and your bid price.** This is the same as current behavior. If the market price for Spot Instances rises above your bid price for a running instance, Amazon EC2 will terminate your instance.
- **Changing your bid price.** If you want to change your bid price, you have to create a new launch configuration and associate it with your Auto Scaling group. You cannot update the existing launch configuration.
- **New bid price and running instances.** When you change your bid price by creating a new launch configuration, instances already launched will continue to run as long as the bid price for those running instances is higher than the current market price for Spot Instances.
- **Spot and Auto Scaling instance termination.** Amazon EC2 terminates a Spot Instance when the bid price for that instance falls below the Spot market price. Auto Scaling terminates instances based on a combination of criteria, including the launch configuration it is associated with, length of time in a billing hour the instance has been running, and the Availability Zone in which it is launched. For more information about instance termination in Auto Scaling, see Auto Scaling Instance Termination.
- **Maintaining your Spot Instances.** When your instance is terminated, Auto Scaling will try to launch another instance to replace it in order to maintain your specified desired capacity. However, whether or not Auto Scaling successfully launches an instance depends on the bid price as compared to the Spot market price: If the bid price is higher than the Spot market price, then an instance will be launched; if the Spot market price is higher than the bid price, then no instance will be launched at that point.
- **Persistent bids.** Auto Scaling will continue submitting bids for you as long as you keep your Auto Scaling group and launch configuration. The Auto Scaling group specifies the desired number of instances you want maintained and the launch configuration specifies the bid price for your Spot Instances.

For more information about Auto Scaling, see Auto Scaling Developer Guide. To get started quickly with Auto Scaling, see Get Started with Auto Scaling.

Tools for Managing Auto Scaling with Spot Instances

You will use the Auto Scaling command line interface (CLI) tools to use the Auto Scaling features with Spot Instances. Currently, the AWS Management Console does not have support to create and manage Auto Scaling objects. However, you can view most details about bids and Amazon EC2 instances launched by Auto Scaling on the AWS Management Console.

To use the Auto Scaling CLI tools, download and unzip the package, and set up the tools on your computer just as you set up your Amazon EC2 CLI tools. The Auto Scaling CLI tools and the Amazon EC2 CLI tools are shipped as different tools packages.

The following table lists resources that can help you get started with the Auto Scaling CLI tools. For more information, go to Auto Scaling Tools in the *Auto Scaling Developer Guide.*

Resources for Using Auto Scaling CLI Tools

Task	Resource
Download the Auto Scaling command line tools	Auto Scaling Command Line Tool on the Amazon Web Services site.
Setup and Install Auto Scaling command line tools	Using the Command Line Tools in the *Auto Scaling Developer Guide.*
Read the readme installation instructions	*Install Drive\AutoScaling-n.n.nn.n*README

To get help about Auto Scaling CLI commands

After you have installed the Auto Scaling CLI tools, you can get more information about the commands by querying the command line.

1. Open the Command Line.
2. Navigate to *Install Drive\AutoScaling-n.n.nn.n*bin.
3. Type `as-cmd` and press **Enter**.

   The command line returns a list of Auto Scaling CLI commands and a short description of what each command does.

   For example, here are some of the Auto Scaling CLI commands we will use in this section.

Command	Description
`as-create-launch-config`	Creates a new launch configuration with specified attributes.
`as-create-auto-scaling-group`	Creates a new Auto Scaling group with specified name and other attributes.
`as-describe-auto-scaling-groups`	Describes the Auto Scaling groups, if the groups exist.
`as-describe-auto-scaling-instances`	Describes the Auto Scaling instances, if the instances exist.

Command	Description
`as-describe-scaling-activities`	Describes a set of activities or all activities belonging to a group.
`as-delete-auto-scaling-group`	Deletes the specified auto scaling group, if the group has no instances and no scaling activities are in progress.

For common conventions the documentation uses to represent command symbols, see Document Conventions.

4. Type `as-command-name --help`.

   The command line returns a description, syntax information, and examples of how to use the specified Auto Scaling CLI command.

For Amazon EC2 CLI tools, see the following:

- **Amazon EC2 CLI tools.** Amazon Elastic Compute Cloud CLI Reference in the *Amazon EC2 Command Line Reference*.
- **Amazon EC2 CLI tools specific to Spot Instances.** Command Line Tools (p. 128) in the *Amazon EC2 User Guide*.
- **Other Amazon EC2 tools you can use with Spot instances.** Get Started with Spot Instances (p. 126) in the *Amazon EC2 User Guide*.

Launch Spot Instances with Auto Scaling

In this section, we will create an Auto Scaling launch configuration and an Auto Scaling group that launch Spot Instances. We will use the Auto Scaling command line interface (CLI) commands to create the launch configuration and the Auto Scaling group, and to verify and obtain information about them and the Amazon EC2 instances that they launch.

Prerequisites

If you're not familiar with how to create a launch configuration or an Auto Scaling group, we recommend that you go through the steps in the Basic Scenario in Auto Scaling in the *Auto Scaling Developer Guide*. Use the basic scenario to get started with the infrastructure that you need in most Auto Scaling scenarios.

If you don't have the Auto Scaling CLI tools installed on your computer, you must install them to do this walkthrough. For information, go to Auto Scaling Tools You Will Use in the *Amazon EC2 User Guide*. You can also use the information in Using the Command Line Tools in the *Auto Scaling Developer Guide*.

In this scenario, we will perform the following tasks:

- Step 1: Create a Launch Configuration (p. 147)
- Step 2: Create an Auto Scaling Group (p. 148)

If you already have a launch configuration and Auto Scaling group, here are other related Spot Instance tasks that you can do using Auto Scaling:

- Update the Bid Price for the Spot Instances (p. 153)
- Schedule Spot Bid Requests (p. 156)
- Get Notifications through Auto Scaling for Spot Instances (p. 160)

- Advanced Tasks (p. 163)

For more information about Auto Scaling, see Understanding Auto Scaling in the *Auto Scaling Developer Guide*. For information about scenarios using Auto Scaling, see Using Auto Scaling also in the *Auto Scaling Developer Guide*.

Step 1: Create a Launch Configuration

An Auto Scaling launch configuration is a template that contains the parameters necessary to launch new Amazon EC2 instances. You can attach only one launch configuration to an Auto Scaling group at a time. You have to create a launch configuration and then an Auto Scaling group in order to launch instances with Auto Scaling.

To place bids for Spot Instances, use the `as-create-launch-config` Auto Scaling CLI command with the `--spot-price` option. Specify the maximum price you want to pay for an instance. This price will be used by Auto Scaling to bid for your instances, but this price is not necessarily what you pay for the instances when they are launched. You will pay the Spot price. For example, you bid $0.05 for m1.small instances. Your bid gets fulfilled if the current market price for m1.small Spot Instance is $0.03, or any other price below $0.05, and you will be charged the current price of $0.03 per hour. In fact, as long as your current bid is higher than the market price, your bid will be fulfilled and a Spot Instance will be launched for you.

You can get guidance on the price to bid by checking Spot price history. You can do this by using the AWS Management Console, CLI, or API. For more information, go to View Spot Price History (p. 129).

The `as-create-launch-config` command takes the following arguments:

```
as-create-launch-config LaunchConfigurationName --image-id value --instance-type
value [--spot-price value] [--iam-instance-profile value] [--block-device-mapping
"key1=value1,key2=value2..." ] [--monitoring-enabled|--monitoring-disabled]
[--kernel value ] [--key  value ] [--ramdisk value] [--group value[,value...]
] [--user-data value] [--user-data-file value] [General Options]
```

The only required options are the launch configuration name, image ID, and instance type. For this walkthrough, specify:

- Launch configuration name = `spotlc-5cents`

  Note

  When Auto Scaling launches instances, it does not distinguish the Spot Instances from the On-Demand instances. To help you identify which of your instances are Spot Instances, consider assigning a launch configuration name that includes *spot* and the bid price.

- Image ID = `ami-e565ba8c`
  The Image ID identifies the Amazon Machine Image (AMI) that will be used to launch the instances. If you don't have an AMI, and you want to find a suitable one, see Amazon Machine Images (AMIs).

- Instance type = `m1.small`

- Spot price = `$0.05`

  This parameter is optional. If you want to launch Spot Instances, you must specify the Spot bid price that will be used to bid for Spot Instances. Spot Instance bid prices must be specified in US dollars.

Your command should look similar to the following example:

```
as-create-launch-config spotlc-5cents --image-id ami-e565ba8c --instance-type
m1.small --spot-price "0.05"
```

You should get a confirmation like the following example:

```
OK-Created launch config
```

Step 2: Create an Auto Scaling Group

An Auto Scaling group is a collection of Amazon EC2 instances that shares similar characteristics and to which you want to apply certain scaling actions. You can use the Auto Scaling group to automatically scale the number of instances or maintain a fixed number of instances. You can attach only one launch configuration to an Auto Scaling group at a time. You have to create a launch configuration and then an Auto Scaling group in order to launch Amazon EC2 instances with Auto Scaling.

The `as-create-auto-scaling-group` command takes the following arguments:

```
as-create-auto-scaling-group AutoScalingGroupName --availability-zones
value[,value...] --launch-configuration value --max-size value --min-size value
[--default-cooldown value] [--desired-capacity value] [--grace-period value]
[--health-check-type value] [--load-balancers value[, value]] [--placement-group
value] [--vpc-zone-identifier value] [General Options]
```

This command requires that you specify a name for your Auto Scaling group, a launch configuration, one or more Availability Zones, a minimum group size, and a maximum group size. The Availability Zones you choose determine the physical location of your Auto Scaling instances. The minimum and maximum group size tells Auto Scaling the minimum and maximum number of instances the Auto Scaling group should have.

Desired capacity is an important component of the `as-create-auto-scaling-group` command. Although it is an optional parameter, desired capacity tells Auto Scaling the number of instances you want to run initially. To adjust the number of instances you want running in your Auto Scaling group, you change the value of `--desired-capacity`. If you don't specify `--desired-capacity`, its value is the same as minimum group size.

For more detailed information on the syntax of the `as-create-auto-scaling-group` command, see Create an Auto Scaling Group in the *Auto Scaling Developer Guide*. You can also get help information from the command line: Run the `as-create-auto-scaling-group --help`. For more information, go to Resources for Using Auto Scaling CLI Tools (p. 145).

For this walkthrough, specify these values for the command:

- Auto Scaling Group name = `spotasg`

  Note

  When Auto Scaling launches instances, it does not distinguish the Spot Instances from the On-Demand instances. To help you identify which of your instances are Spot Instances, consider assigning spot-specific names to the Auto Scaling group that launches Spot Instances.

- Launch configuration name = `spotlc-5cents`
- Availability Zone = `us-east-1a,us-east-1b`
- Max size = `5`
- Min size = `1`
- Desired capacity = `3`

Your command should look like the following example:

```
as-create-auto-scaling-group spotasg --launch-configuration spotlc-5cents --
availability-zones "us-east-1a,us-east-1b" --max-size 5 --min-size 1 --desired-
capacity 3
```

You should get confirmation like the following example:

```
OK-Created AutoScalingGroup
```

What do you want to do next?

- Obtain Information About the Instances Launched by Auto Scaling (p. 149)
- Update the Bid Price for the Spot Instances (p. 153)
- Schedule Spot Bid Requests (p. 156)
- Get Notifications through Auto Scaling for Spot Instances (p. 160)
- Advanced Tasks (p. 163)

For more information about Auto Scaling, see Understanding Auto Scaling in the *Auto Scaling Developer Guide*. For information about scenarios using Auto Scaling, see Using Auto Scaling also in the *Auto Scaling Developer Guide*.

Obtain Information About the Instances Launched by Auto Scaling

You can use the Auto Scaling CLI tools to obtain information about your launch configuration, Auto Scaling groups and Amazon EC2 instances launched by Auto Scaling.

In this section, we will use the following Auto Scaling CLI commands to get information about the Spot price bids and Spot Instances that Auto scaling makes and launches for you.

- `as-describe-scaling-activities`—You can use the information about Auto Scaling activities that this command returns to check the status of the bids submitted for you by Auto Scaling.
- `as-describe-auto-scaling-groups`—You can use the information about Auto Scaling groups that this command returns to confirm that Auto Scaling is launching your Spot Instances according to your specifications.

To check the status of the bids that Auto Scaling is making for you

The `as-describe-scaling-activities` command lists the activities that Auto Scaling performed for a specified Auto Scaling group.

This is the basic syntax:

```
as-describe-scaling-activities [ActivityIds [ActivityIds ...]]
[--auto-scaling-group value] [--max-records value] [General Options]
```

Specifying the Auto Scaling group and the Activity ID are optional. If you don't specify the Auto Scaling group, the command will return all activities for all Auto Scaling groups associated with your account. If you specify the Auto Scaling group, only the activities for that Auto Scaling group will be listed.

In this scenario, we are using the `as-describe-scaling-activities` command to see state of your bid. Assume that there is only one Auto Scaling group (spotasg) and you want to list all activities.

1.   Open a command line and navigate to the bin folder of your Auto Scaling CLI tools directory.

2. Type the command: `as-describe-scaling-activities --auto-scaling-group spotasg --headers`

The information you get should be similar to the following example.

```
ACTIVITY  ACTIVITY-ID                               END-TIME              GROUP-
NAME  CODE                     MESSAGE
ACTIVITY  31bcbb67-7f50-4b88-ae7e-e564a8c80a90                     spotasg
     WaitingForSpotInstanceId  Placed Spot instance request: sir-fc8a3014.
 Waiting for instance(s)
ACTIVITY  770bbeb5-407c-404c-a826-856f65db1c57                     spotasg
     WaitingForSpotInstanceId  Placed Spot instance request: sir-69101014.
 Waiting for instance(s)
ACTIVITY  597e4ebd-220e-42bc-8ac9-2bae4d20b8d7  2012-05-23T17:40:22Z  spotasg
     Successful
```

In this response, you know that your bids were placed, one of the bids is successful, and Auto Scaling is waiting for the other two bids.

Note

If the `as-describe-scaling-activities` command returns a list that includes *Failed* activities, check the data you specified in the launch configuration. For example:

- The Amazon Machine Image (AMI) might not be valid anymore.
- The bid price specified in the launch configuration is lower than the Spot market price.

3. If you run the `as-describe-scaling-activities` command again later, you can be getting information that is similar to the following example:

```
ACTIVITY  ACTIVITY-ID                               END-TIME              GROUP-
NAME       CODE
ACTIVITY  90630906-b40f-41a6-967a-cd6534b2dfca  2012-06-01T02:32:15Z  spotasg
          Successful
ACTIVITY  a1139948-ad0c-4600-9efe-9dab8ce23615  2012-06-01T00:48:02Z  spotasg
          Successful
ACTIVITY  33001e70-6659-4494-a817-674d1b7a2f58  2012-06-01T02:31:11Z  spotasg
          Successful
```

The output shows that the listed activities were successful. Because we know that *spotasg* is an Auto Scaling group that uses a launch configuration with a spot bid price, you can assume that the activities represent the bids placed by Auto Scaling.

4. If you want to get more details about the activities and instances, use the `--show-xml` option of `as-describe-scaling-activities`. Enter the following command
`as-describe-scaling-activities --auto-scaling-group spotasg --show-xml`.

The information you get should be similar to the following example.

```
<DescribeScalingActivitiesResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <DescribeScalingActivitiesResult>
    <NextToken>b5a3b43e-10c6-4b61-8e41-2756db1fb8f5</NextToken>
    <Activities>
      <member>
```

```
        <StatusCode>Successful</StatusCode>
        <Progress>0</Progress>
        <ActivityId>90630906-b40f-41a6-967a-cd6534b2dfca</ActivityId>
        <StartTime>2012-06-01T00:48:21.942Z</StartTime>
        <AutoScalingGroupName>spotasg</AutoScalingGroupName>
        <Cause>At 2012-06-01T00:48:21Z a difference between desired and ac
tual capacity changing the desired capacity, increasing the capacity from
2 to 3.</Cause>
        <Details>{}</Details>
        <Description>Launching a new EC2 instance: i-fe30d187</Description>

        <EndTime>2012-06-01T02:32:15Z</EndTime>
      </member>
      <member>
        <StatusCode>Successful</StatusCode>
        <Progress>0</Progress>
        <ActivityId>a1139948-ad0c-4600-9efe-9dab8ce23615</ActivityId>
        <StartTime>2012-06-01T00:47:51.293Z</StartTime>
        <AutoScalingGroupName>spotasg</AutoScalingGroupName>
        <Cause>At 2012-06-01T00:47:51Z an instance was taken out of service
 in response to a system health-check.</Cause>
        <Details>{}</Details>
        <Description>Terminating EC2 instance: i-88ce28f1</Description>
        <EndTime>2012-06-01T00:48:02Z</EndTime>
      </member>
      <member>
        <StatusCode>Successful</StatusCode>
        <Progress>0</Progress>
        <ActivityId>33001e70-6659-4494-a817-674d1b7a2f58</ActivityId>
        <StartTime>2012-06-01T00:46:19.723Z</StartTime>
        <AutoScalingGroupName>spotasg</AutoScalingGroupName>
        <Cause>At 2012-06-01T00:46:19Z a difference between desired and ac
tual capacity changing the desired capacity, increasing the capacity from
2 to 3.</Cause>
        <Details>{}</Details>
        <Description>Launching a new EC2 instance: i-2c30d155</Description>

        <EndTime>2012-06-01T02:31:11Z</EndTime>
      </member>
      ...
    </Activities>
  </DescribeScalingActivitiesResult>
  <ResponseMetadata>
    <RequestId>d02af4bc-ad8f-11e1-85db-83e1968c7d8d</RequestId>
  </ResponseMetadata>
</DescribeScalingActivitiesResponse>
```

The XML output shows more detail about the Spot and Auto Scaling activity.

- `Cause: At 2012-06-01T00:48:21Z a difference between desired and actual`
  `capacity changing the desired capacity, increasing the capacity from 2 to`
  `3. Description: Launching a new EC2 instance: i-fe30d187`

  If an instance is terminated and the number of instances falls below the desired capacity, Auto
  Scaling will launch a new instance so that the total number of your running instances rises back
  to the level specified for desired capacity.

- Cause: At 2012-06-01T00:47:51Z an instance was taken out of service in response to a system health-check. Description: Terminating EC2 instance: i-88ce28f1

Auto Scaling maintains the desired number of instances by monitoring the health status of the instances in the Auto Scaling group. When Auto Scaling receives notification that an instance is *unhealthy* or terminated, Auto Scaling launches another instance to take the place of the unhealthy instance. For information about how Auto Scaling monitors the health status of instances, go to Maintaining Current Scaling Level in the *Auto Scaling Developer Guide*.

Note

Auto Scaling provides the cause of instance termination that is not the result of a scaling activity. This includes instances that have been terminated because the Spot market price exceeded their bid price.

When Auto Scaling attempts to replace terminated instances resulting from the Spot market price rising above the instances' bid price, Auto Scaling will place the bid specified in the current launch configuration and attempt to launch another instance to maintain the desired capacity.

To confirm that Auto Scaling is launching your Spot Instances according to your specifications

Use `as-describe-auto-scaling-groups`. The command will show details about the group and instances launched. For information about the `as-describe-auto-scaling-groups` command, see Verify Auto Scaling Group Creation in the *Auto Scaling Developer Guide*.

1. Open a command line and navigate to the bin folder of your Auto Scaling CLI tools directory.
2. Type the command:`as-describe-auto-scaling-groups spotasg --headers`

   Note

   The `--headers` option supplies the column name so you know what data is being returned.

   The information you get should be similar to the following example.

   ```
   AUTO-SCALING-GROUP   GROUP-NAME        LAUNCH-CONFIG   AVAILABILITY-ZONES
   MIN-SIZE   MAX-SIZE   DESIRED-CAPACITY
   AUTO-SCALING-GROUP   spotasg           spotlc-5cents   us-east-1b,us-east-1a
   1          5          3
   INSTANCE   INSTANCE-ID   AVAILABILITY-ZONE   STATE        STATUS    LAUNCH-CONFIG
   INSTANCE   i-2c30d155    us-east-1a          InService    Healthy   spotlc-5cents
   INSTANCE   i-fe30d187    us-east-1a          InService    Healthy   spotlc-5cents
   INSTANCE   i-c630d1bf    us-east-1a          InService    Healthy   spotlc-5cents
   ```

   You can see that Auto Scaling launched 3 instances in us-east-1a, as you specified, and they are all running.
3. Additionally, you can find out details about the Spot Instances launched for you by Auto Scaling, by using the `as-describe-auto-scaling-instances` command.

   This is the basic syntax:

   ```
   as-describe-auto-scaling-instances [InstanceIds [InstanceIds ...]]
   [--max-records value] [General Options]
   ```

Specifying `InstanceIds` is optional. If you specify it, the command will return information about the instance, if it exists. If you don't specify `InstanceIds`, the command returns information about all instances associated with your Auto Scaling account.

In this walkthrough, we are assuming that you created one launch configuration and Auto Scaling group, and you want to find out details about all your Spot Instances.

Your command should look like the following example:

```
as-describe-auto-scaling-instances --headers
```

The information you get should be similar to the following example:

```
INSTANCE   INSTANCE-ID  GROUP-NAME      AVAILABILITY-ZONE  STATE      STATUS
   LAUNCH-CONFIG
INSTANCE  i-2c30d155   spotasg         us-east-1a         InService  HEALTHY
  spotlc-5cents
INSTANCE  i-c630d1bf   spotasg         us-east-1a         InService  HEALTHY
  spotlc-5cents
INSTANCE  i-fe30d187   spotasg         us-east-1a         InService  HEALTHY
  spotlc-5cents
```

What do you want to do next?

- Update the Bid Price for the Spot Instances (p. 153)
- Schedule Spot Bid Requests (p. 156)
- Get Notifications through Auto Scaling for Spot Instances (p. 160)
- Advanced Tasks (p. 163)

Update the Bid Price for the Spot Instances

Auto Scaling launch configurations cannot be changed. If you want to modify your bid price for Spot Instances, you must create a new launch configuration.

If, for example, you want to launch a set of Spot Instances that have a higher likelihood of running uninterrupted for a long time, you can use a higher bid price. To do this, you must create a new launch configuration, using the same procedure that you followed earlier in this walkthrough. (For more information, go to Step 1: Create a Launch Configuration (p. 147).)

Specify the following values:

- Launch configuration name = spotlc-7cents
- Image ID = ami-e565ba8c

> **Note**
>
> If you don't have an AMI, and you want to find a suitable one, see Amazon Machine Images (AMIs).

- Instance type = m1.small
- Spot price = $0.07

Your command should look similar to the following example:

```
as-create-launch-config spotlc-7cents --image-id ami-e565ba8c --instance-type
m1.small --spot-price "0.07"
```

You should get a confirmation like the following example:

```
OK-Created launch config
```

After you have created the new launch configuration successfully, create a new Auto Scaling group
specifying the new launch configuration.

Your command should look similar to the following example:

```
as-create-auto-scaling-group spotasg-7cents --launch-configuration spotlc-7cents
 --availability-zones "us-east-1a,us-east-1b" --max-size 5 --min-size 10 --de
sired-capacity 3
```

You should get a confirmation like the following example:

```
OK-Created AutoScalingGroup
```

You can view the status of your Spot bid and a list of the bids that Auto Scaling placed for you by running
as-describe-scaling-activities soon after you create your Auto Scaling group.

Your command should look similar to the following example:

```
as-describe-scaling-activities --headers
```

If not all your bids are fulfilled, you will get information that looks like the following example:

```
ACTIVITY   ACTIVITY-ID                                 END-TIME                GROUP-
NAME      CODE                       MESSAGE
ACTIVITY   5879cc50-1e40-4539-a754-1cb084f1aecd                         spotasg-
7cents  WaitingForSpotInstanceId  Placed Spot instance request: sir-93828812.
Waiting for instance(s)
ACTIVITY   777fbe1b-7a24-4aaf-b7a9-d368d0511878                         spotasg-
7cents  WaitingForSpotInstanceId  Placed Spot instance request: sir-016cf812.
Waiting for instance(s)
ACTIVITY   f4b00f81-eaea-4421-80b4-a2e3a35cc782                         spotasg-
7cents  WaitingForSpotInstanceId  Placed Spot instance request: sir-cf60ea12.
Waiting for instance(s)
ACTIVITY   31bcbb67-7f50-4b88-ae7e-e564a8c80a90                          spotasg
        WaitingForSpotInstanceId  Placed Spot instance request: sir-fc8a3014.
Waiting for instance(s)
ACTIVITY   770bbeb5-407c-404c-a826-856f65db1c57                          spotasg
        WaitingForSpotInstanceId  Placed Spot instance request: sir-69101014.
Waiting for instance(s)
ACTIVITY   597e4ebd-220e-42bc-8ac9-2bae4d20b8d7  2012-05-23T17:40:22Z  spotasg
        Successful

ACTIVITY   eca158b4-a6f9-4ec5-a813-78d42c1738e2  2012-05-23T17:40:22Z  spotasg
        Successful
```

```
ACTIVITY   1a5bd6c6-0b0a-4917-8cf0-eee1044a179f   2012-05-23T17:22:19Z   spotasg
         Successful

ACTIVITY   c285bf16-d2c4-4ae8-acad-7450655facb5   2012-05-23T17:22:19Z   spotasg
         Successful

ACTIVITY   127e3608-5911-4111-906e-31fb16d43f2e   2012-05-23T15:38:06Z   spotasg
         Successful

ACTIVITY   bfb548ad-8bc7-4a78-a7db-3b41f73501fc   2012-05-23T15:38:07Z   spotasg
         Successful

ACTIVITY   82d2b9bb-3d64-46d9-99b6-054a9ecf5ac2   2012-05-23T15:30:28Z   spotasg
         Successful

ACTIVITY   95b7589b-f8ac-49bc-8c83-514bf664b4ee   2012-05-23T15:30:28Z   spotasg
         Successful

ACTIVITY   57bbf77a-99d6-4d94-a6db-76b2307fb9de   2012-05-23T15:16:34Z   spotasg
         Successful

ACTIVITY   cdef758e-0be2-416e-b402-0ef521861039   2012-05-23T15:16:17Z   spotasg
         Successful

ACTIVITY   d7e0a3ed-7067-4583-8a87-1561b3de2aed   2012-05-23T14:51:46Z   spotasg
         Successful

ACTIVITY   da5471ab-482c-4680-b430-99e4173d2bd7   2012-05-23T14:52:48Z   spotasg
         Successful

ACTIVITY   78701f3d-a747-46e1-8b0f-8aff22834f46   2012-05-23T14:38:17Z   spotasg
         Successful

ACTIVITY   274d4772-3614-4f5c-8858-026b33635be3   2012-05-23T14:38:16Z   spotasg
         Successful

ACTIVITY   1024abb2-bf84-4fae-b717-a398bac91c4f   2012-05-23T14:22:39Z   spotasg
         Successful
```

Bids are represented by values such as `sir-93828812` and `sir-016cf812`.

When you create a new launch configuration that sets a new bid price for Spot Instances, and you have Spot Instances already running based on a different price, these instances will continue running and will only be terminated if the Spot market price goes above the bid price on which it was based.

What do you want to do next?

- Schedule Spot Bid Requests (p. 156)
- Get Notifications through Auto Scaling for Spot Instances (p. 160)
- Advanced Tasks (p. 163)

Schedule Spot Bid Requests

You can set up Auto Scaling to launch a certain number of instances at a specific time. This capability is useful if, for example, you want to take advantage of a window of time when prices historically are lower, or you want to terminate Spot Instances at a specific time.

We will use the Auto Scaling CLI command `as-put-scheduled-update-group-action` to set up a schedule. This is the basic syntax:

as-put-scheduled-update-group-action *ScheduledActionName* --auto-scaling-group *value* [--desired-capacity *value*] [--end-time *value*][--max-size *value*][--min-size *value*] [--recurrence *value*][--start-time *value*][--time *value*][General Options]

In this scenario, use the following values:

- Scheduled action name: `as-spotbid-schedule`
- Auto Scaling group: `spotasg`
- Start time: `2012-05-15T19:10:00Z`
- End time: `2012-05-15T19:15:00Z`
- Desired capacity:`20`

Your command should look similar to the following example:

```
as-put-scheduled-update-group-action as-spotbid-schedule --auto-scaling-group
spotasg --desired-capacity 20 --start-time 2012-05-15T19:10:00Z --end-time 2012-
05-15T19:15:00Z
```

You should get a confirmation like the following example:

```
OK-Put Scheduled Update Group Action
```

To check your scheduled action, run `as-describe-scheduled-actions`.

You will get information similar to the following example:

```
UPDATE-GROUP-ACTION spotasg as-spotbid-schedule 2012-05-15T19:10:00Z 20
```

What do you want to do next?

Using CloudFormation Templates to Launch Spot Instances

You can use AWS CloudFormation templates to launch Spot Instances and the AWS resources you need for an application. Templates are text files describing a collection, called *stack*, of AWS resources that you want to deploy as a group. For more information about AWS CloudFormation templates, see Learn Template Basics. For more information about using AWS CloudFormation, see the AWS CloudFormation User Guide.

You can write your own template from scratch, or you can use one of the example templates from the AWS CloudFormation Sample Template Library. The following templates in AWS CloudFormation utilize Spot Instances:

- Asynchronous Queue-Based Processing

  This template creates an auto-scaled worker that monitors work (messages) in a queue. The application is auto-scaled based on the amount of work in the queue. When there is work, Auto Scaling scales up; when there is no work, Auto Scaling scales down. Each message contains a command/script to run, an input file location, and an output location for the results.

  To download this template, go to Asynchronous Queue-Based Processing.
- Bees with Machine Guns

  This template creates a load balancer, a controller, and the instances behind the load balancer; fires up and riggers an attack; and stores logs on Amazon Simple Storage Service (Amazon S3) and then shuts down (if enabled). For this template, instances use the Amazon Linux AMI, and the setup requires that you already have an AWS Identity and Access Management (IAM) SSL certification. You can modify this template to remove the SSL dependencies.

  Note

  To launch this template, you need a new SSH private key that you create specifically for this task. You'll have to log in to the instance created by the template and provide this private key information.

  To download this template, go to Bees with Machine Guns.
- Grid Computing for Spot Instances

  This template uses Star Cluster to launch and bootstrap a cluster of Amazon EC2 instances for high performance computational tasks using Spot pricing. You only need to provide the number of instances, instance size, and Spot price that you want to use.

  To download this template, go to Grid Computing for Spot Instances.

You can create stacks from these templates by using the AWS Management Console, the AWS CloudFormation command line tools, or the AWS CloudFormation API.

Before you use these templates, you must:

1. Have an AWS account and sign up for AWS CloudFormation.
2. Decide on the template to use.
3. Enter the parameters required for the stack.
4. Create the stack.

To get started with AWS CloudFormation, Getting Started with AWS CloudFormation.

For more information about using AWS CloudFormation, see the AWS CloudFormation User Guide.

Launching Amazon Elastic MapReduce Job Flows with Spot Instances

You can launch an Amazon Elastic MapReduce job flow in Spot Instances. Amazon Elastic MapReduce is a data analysis tool that simplifies the set up and management of a computer cluster, the source data, and the computational tools that help you implement sophisticated data processing jobs quickly. For more information, see Introduction to Amazon Elastic MapReduce.

Quick Look: Using Spot Instances with Amazon Elastic MapReduce Video

The following video describes how Spot Instances work in Amazon EMR and walks you through the process of launching a job flow on Spot Instances from the AWS Management Console: Using Spot Instances with Amazon ElasticMapReduce

Launch Spot Instances in Amazon Virtual Private Cloud

Amazon Virtual Private Cloud (Amazon VPC) lets you define a virtual networking environment in a private, isolated section of the Amazon Web Services (AWS) cloud. You have complete control of this virtual network and you can use advanced security features and network access control at the instance level and subnet level. For more information about Amazon VPC, go to the Amazon Virtual Private Cloud User Guide.

If you want to take advantage of the features of Amazon VPC when you use Spot Instances, specify in your Spot Request that your instances are to be launched in Amazon VPC. Before you can specify that your Spot Instances are to be launched in Amazon VPC, you must set up a VPC. For information about setting up an Amazon VPC, see Getting Started with Amazon VPC in the *Amazon Virtual Private Cloud Getting Started Guide*.

### Quick Look: Launching Spot Instances in Amazon VPC Video

The following video shows how to launch your first Spot Instance in the Amazon Virtual Private Cloud (Amazon VPC) using the AWS Management Console. This video includes instructions for creating your Amazon VPC subnet, placing a bid, determining when the instance is fulfilled, and canceling the instance. Launching Spot Instances in Amazon VPC Video

The process for making a Spot Instance request that launches in Amazon VPC is the same as the process you follow when you make a Spot Instance request in a non-VPC portion of Amazon EC2. The main differences are that you:

- Base your Spot Price bid on Spot Price history of Spot Instances in VPCs.
  When you use the `DescribeSpotPriceHistory` action or the `ec2-describe-spot-price-history` command, add *Amazon VPC* to the `product-description` filter. For example:

```
PROMPT> ec2-describe-spot-price-history -s '2011-09-08T00:00:00Z' -t m1.xlarge
 -d "Linux/UNIX (Amazon VPC)"
```

Using the AWS Management Console, check the **Spot Price History** page to see the Spot pricing history for Amazon EC2 instances running in both Amazon EC2 and Amazon VPC.
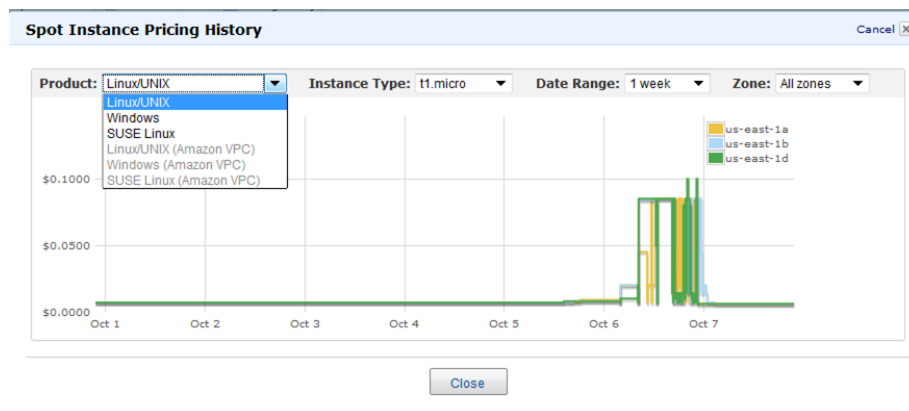
- Specify the VPC subnet in which you want to launch your Spot Instance.

When you use the `RequestSpotInstances` action or the `ec2-request-spot-instances` command, specify the ID of the Amazon VPC subnet in which you want to launch the Spot Instance.

```
PROMPT> ec2-request-spot-instances ami-8e1fece7 -t m1.xlarge -p '$0.01' -n 5
 -r 'one-time' -s 'subnet-baab943d3'
```

When you launch the **Request Instances Wizard** from the Spot Instance page of the AWS Management Console, select a subnet after specifying that you're launching the Spot Instance into a VPC.

Note

The **Request Spot Instances** page will show the **VPC** option only if you have created a VPC and subnet before making the Spot Instance request.

For more information about using Amazon VPC, go to the Amazon Virtual Private Cloud User Guide.

Plan for Interruptions

Because Spot Instances can terminate at any time, applications that run on Spot Instances must terminate cleanly. Although we attempt to cleanly terminate your instances, your application should be prepared to deal with an immediate shutdown.

To test your application, launch and terminate it as an on-demand instance.

Important

Although Spot Instances can use Amazon EBS-backed AMIs, be aware that the EBS-backed AMIs don't support Stop/Start. In other words, you can't stop and start Spot Instances launched from an AMI with an Amazon EBS root device.

Quick Look: Managing Interruptions Video

The following video shows how some customers manage the interruption of their Spot instances. How to Manage Spot Instance Interruptions

For more information about strategies to manage interruptions, go to the following sections:

- Launching Amazon Elastic MapReduce Job Flows with Spot Instances (p. 157)

Get Notifications through Auto Scaling for Spot Instances

You can set up Auto Scaling to send notifications to you as instances are terminated and launched. When the Spot market price goes up and your bid price falls below it, Amazon EC2 terminates the instances that were launched based on that bid price. If your Spot Instances are terminated, Auto Scaling will try to submit your bid and launch replacement instances and to ensure the capacity you specified for your Auto Scaling group. You can set up Auto Scaling to notify you when instance launch or termination events occur.

There are two ways to get notifications for Spot Instances:

- Auto Scaling
- AWS SDK Sample Code Tool

Spot Notifications Sample in AWS Java SDK

You can also use the AWS Java SDK to develop applications that monitor your Spot Instance usage in ways that are customized to your use case. The Spot Notifications sample application is a Java application that demonstrates techniques for monitoring Amazon EC2 instance status, Spot Instance requests, and Spot Price fluctuations. The application is documented and freely available for download at How to Track Spot Instance Activity with the Spot-Notifications Sample Application. You are free to adapt the application for your own purposes, or use it as a guide in developing your own application for monitoring Spot Instances. For more information, go to the AWS SDK for Java.

Configuring Auto Scaling groups to send notifications about your Spot Instances

In this portion of the walkthrough, you learn how to set up Amazon SNS to send email notifications.

To do this, you need the following:
An Amazon Resource Name (ARN). You generate an ARN when you create an Amazon Simple Notification Service (Amazon SNS) topic. A topic is a communication channel to send messages and subscribe to notifications. It provides an access point for publishers and subscribers to communicate with each other. An endpoint, such as an email address, must be subscribed to the topic so the endpoint can receive messages published to the topic. To create a topic, go to Create a Topic in the *Amazon Simple Notification Service Getting Started Guide*.

An Auto Scaling Group. Use the Auto Scaling group that you created earlier in this walkthrough.

A notification configuration. You configure an Auto Scaling group to send notifications when specified events take place by calling the `as-put-notification-configuration` CLI command or the `PutNotificationConfiguration` API action. We will discuss the steps in setting up a notification configuration later in this section. For more information about the command, go to PutNotificationConfiguration in the *Auto Scaling API Reference*.

A list of Auto Scaling notification types. These notification types are the events that will cause the notification to be sent.

Set Up Notifications Using Auto Scaling

1. After you've created your Amazon SNS topic and you have the ARN, you are ready to set up the notification configuration. (To create a topic, go to Create a Topic in the *Amazon Simple Notification Service Getting Started Guide*.)

   To configure your Auto Scaling group to send notifications when specified events take place, use the `as-put-notification-configuration` CLI command.

The `as-put-notification-configuration` command takes the following arguments:

as-put-notification-configuration *AutoScalingGroupName* **--notification-types**
value **--topic-arn** *topic-ARN* **[General Options]**

You need to specify the Auto Scaling group name, the ARN, and the notifcation types.

For this example, specify:

- Auto Scaling group name: `spotasg`
  Specify the Auto Scaling group that you want to get notifications about. In this walkthrough, you want Auto Scaling to notify you when instances are launched and terminated for the *spotasg* Auto Scaling group.

- ARN:`arn:placeholder:MyTopic`

    Note

    ARNs are unique identifiers for Amazon Web Services (AWS) resources. Replace the ARN placeholder with your ARN.

- Notification types: `autoscaling:EC2_Instance_Launch`,
  `autoscaling:EC2_Instance_Terminate`
  The notification types are the events you want Auto Scaling to send you e-mail about.

Open a command prompt and enter the `as-put-notification-configuration` command.

```
as-put-notification-configuration spotasg--topic-arn arn:placeholder:MyTopic
 --notification-types autoscaling:EC2_INSTANCE_LAUNCH, autoscaling:EC2_IN
STANCE_TERMINATE
```

Auto Scaling returns the following:

```
OK-Put Notification Configuration
```

You now have a notification configuration that sends a notification to the endpoint subscribed in the `arn:placeholder:MyTopic` ARN whenever instances are launched or terminated.

2. Verify the notification configuration.

   To verify the notification actions associated with your Auto Scaling group when specified events take place, use `as-describe-notification-configurations`.

   The `as-describe-notification-configurations` command takes the following arguments:

   as-describe-notification-configurations [--auto-scaling-groups
   value[*,value...*]] **[--maxrecords** *value*] **[General Options]**

   If you specify the Auto Scaling group, this command returns a full list of all notification configuration for the Auto Scaling group listed. If you don't provide an Auto Scaling group name, the service returns the full details of all Auto Scaling groups. The command also returns a token if there are more pages to retrieve. To get the next page, call this action again with the returned token as the `next-token` argument. For this walkthrough, specify:

   - Auto Scaling group name: `spotasg`

   Open a command prompt and enter the `as-describe-notification-configurations` command.

   ```
as-describe-notification-configurations --auto-scaling-groups spotasg -headers
```

Auto Scaling returns the following:

```
NOTIFICATION-CONFIG   GROUP-NAME   TOPIC-ARN   NOTIFICATION-TYPE-NAME
NOTIFICATION-CONFIG   spotasg   arn:placeholder:spotasg   autoscaling:EC2_IN
STANCE_LAUNCH
NOTIFICATION-CONFIG   spotasg   arn:placeholder:spotasg   autoscaling:EC2_IN
STANCE_TERMINATE
```

You have confirmed that you have a notification configuration set up for the `spotasg` Auto Scaling group.

When Auto Scaling launches instances to reach or maintain desired capacity, as specified in your Auto Scaling group, SNS sends a notification to your email address with information about the instances. You can check your email Inbox for this information, then run `as-describe-auto-scaling-group` to get information about current instances in the Auto Scaling group and confirm that the instances listed in your email actually exist.

What do you want to do next?

- Obtain Information About the Instances Launched by Auto Scaling (p. 149)
- Update the Bid Price for the Spot Instances (p. 153)
- Schedule Spot Bid Requests (p. 156)
- Advanced Tasks (p. 163)

Starting Clusters on Spot Instances

Grids are a form of distributed computing that enable a user to leverage multiple instances to perform parallel computations. Customers—such as Numerate, Scribd, and the University of Barcelona/University of Melbourne—use Grid Computing with Spot Instances because this type of architecture can take advantage of Spot Instance's built-in elasticity and low prices to get work done faster at a more cost-effective price.

To get started, a user will break down the work into discrete units called jobs, and then submit that work to a "master node." These jobs will be queued up, and a process called a "scheduler" will distribute that work out to other instances in the grid, called "worker nodes." After the result is computed by the worker node, the master node is notified, and the worker node can take the next operation from the queue. If the job fails or the instance is interrupted, the job will automatically be re-queued by the scheduler process.

As you work to architect your application, it is important to choose the appropriate amount of work to be included in your job. We recommend breaking your jobs down into a logical grouping based on the time it would take to process. Typically, you will want to create a workload size less than an hour, so that if you have to process the workload again, it doesn't cost you additional money (you don't pay for the hour if we interrupt your instance).

Many customers use a Grid scheduler, such as Oracle Grid Engine or UniCloud, to set up a cluster. If you have long-running workloads, the best practice is to run the master node on On-Demand or Reserved Instances, and run the worker nodes on Spot or a mixture of On-Demand, Reserved, and Spot Instances. Alternatively, if you have a workload that is less than an hour or you are running a test environment, you may want to run all of your instances on Spot. No matter the setup, we recommend that you create a script to automatically re-add instances that may be interrupted. Some existing tools—StarCluster, for example— can help you manage this process.

Quick Look: How to Launch a Cluster on Spot Video

Chris Dagdigian, from AWS Solution Provider BioTeam, provides a quick overview of how to start a cluster from scratch in about 10 to 15 minutes on Amazon EC2 Spot Instances using StarCluster. StarCluster is an open source tool created by a lab at MIT that makes it easy to set up a new Oracle Grid Engine cluster. In this video, Chris walks through the process of installing, setting up, and running simple jobs on a cluster. Chris also leverages Spot Instances, so that you can potentially get work done faster and potentially save between 50 percent to 66 percent. How to Launch a Cluster on Spot

Bidding Strategies

Depending on how soon you want your Spot Instance request fulfilled and how long you want your Spot Instances to continue running, there are a number of factors you need to consider when you make a bid for Spot Instances. The requirements you include in your Spot Instance request can affect the chances that it will be fulfilled. If you have a number of requirements that must be met, your request can take a little longer to fulfill.

The maximum Spot Instance price impacts bid fulfillment and how long your Spot Instances run, as well. If you want your instances to run continuously for longer periods, select a higher price. If your instances don't need to run continuously, select a lower price.

The maximum Spot Instance price you specify is not necessarily the price that you pay. For example, if you specify $.50 as your maximum price, and the Spot Price is $.30 for the period, you pay $.30. If the Spot Price drops, you pay less. If the Spot Price increases, you pay the new price (up to your specified maximum price).

Quick Look: Bidding Strategies Video

The following video describes strategies to use when bidding for Spot Instances. Deciding on Your Spot Bidding Strategies

Advanced Tasks

Topics
- Persist Your Root EBS Partition (p. 163)
- Tag Spot Instance Requests (p. 164)
- Subscribe to Your Spot Instance Data Feed (p. 165)
- Programming Spot with AWS Java SDK (p. 168)

Now that you have created Spot Instance requests and worked with Spot Instances, this section discusses some advanced tasks.

Persist Your Root EBS Partition

Amazon Elastic Block Store (Amazon EBS) can be an effective way to store data that you otherwise might lose when your Spot Instance terminates.

To set up the persistence of Spot Instance data, you map the Spot Instances that will be launched to an existing Amazon Elastic Block Store (Amazon EBS) snapshot. Set the `delete-on-termination` flag to `false`; this indicates that Amazon EC2 shouldn't delete the Amazon EBS volume when the spot instance terminates.

Let's walk through making an example Spot Request with the following specifications:

- Bid price of $0.5

- One instance of the m1.xlarge instance type
- Block device mapping to a snapshot that shouldn't be deleted when the Spot Instance is terminated

You can do this example using either the CLI or API tools. Using the CLI, your example request should look like this:

```
PROMPT> ec2-request-spot-instances -p 0.5 -t m1.xlarge -n 1 -b '/dev/sdb=snap-
a123bcde:20:false' ami-8e1fece7
```

For more information, see:

- Block Device Mapping (p. 488)
- ec2-request-spot-instances
- RequestSpotInstances

Tag Spot Instance Requests

To help categorize and manage your Spot Instance requests, you can tag them with metadata of your choice. You tag your Spot Instance requests in the same way that you tag other Amazon EC2 resources. Create a tag by specifying a key and value, and assigning the pair to the Spot Instance request. You use the CreateTags action or the ec2-create-tags command. For information about how to use tags, see Tagging Your Resources (p. 508).

The tags you create for your Spot Instance requests only apply to the requests. These tags don't propagate to the instances that the requests launch. To categorize the Spot Instances that are launched from your tagged request, you must create a separate set of tags for the instances using the same commands.

For example, you have Spot Instance request *sir-69047e11* and you want to label it with the tag `Spot=Test`. To do this, use the following command:

```
PROMPT> ec2-create-tags sir-69047e11 --tag "Spot=Test"
```

Amazon EC2 returns this information:

```
TAG     spot-instance-request     sir-69047e11     Spot     Test
```

You can also confirm the tag information by using the `ec2-describe-tags` command.

When your request is fulfilled and a Spot Instance launches, you will see that the tag you used for your Spot Instance request is not applied to the Spot Instance. In the following example command, we are obtaining information about the Spot Instance *i-b8ca48d8* that was launched as a result of your Spot Instance request *sir-69047e11*, tagged Spot=Test.

```
PROMPT> ec2-describe-instances i-b8ca48d8
```

The call returns details about the instance with no tag information, showing that the tag for your Spot Instance request does not apply to the Spot Instance that the request launched. To tag your Spot Instance, use the following command:

```
PROMPT> ec2-create-tags i-b8ca48d8 --tag "SpotI=Test1"
```

Amazon EC2 returns this information:

```
TAG      instance      i-b8ca48d8      SpotI      Test1
```

You can create similar tags using the API. For more information, see the API section of Using Tags in the *Amazon Elastic Compute Cloud User Guide*.

Subscribe to Your Spot Instance Data Feed

If you want to monitor your Spot Instance usage, you can subscribe to your Spot Instance data feed, which stores usage logs in Amazon Simple Storage Service (Amazon S3).

This section describes the data feed content and how to create the data feed for Spot Instances. You can create one data feed per account.

Spot Instance Data Feed Overview

To help you understand the charges for your Spot Instances, Amazon EC2 provides access to a data feed that details your Spot Instance usage and pricing. This data feed is sent to the Amazon S3 bucket of your choice.

To have a data feed delivered to an Amazon S3 bucket, you need to create a Spot Instances data feed subscription using the Amazon EC2 API. When you create this subscription, you can specify an Amazon S3 bucket to deliver the data feed files to, and a filename prefix to use to avoid collisions.

Data Feed Filename and Format

The Spot Instance data feed filename uses the following format (with the date and hour in UTC):

```
{Bucket}.s3.amazonaws.com/{Optional Prefix}/{AWS Account ID}.{YYYY}-{MM}-{DD}-
{HH}.{n}.{Unique ID}.gz
```

For example, if your bucket name is `myawsbucket`, and you name your prefix `myprefix`, your filenames look similar to this:

```
myawsbucket.s3.amazonaws.com/myprefix/111122223333.2010-03-17-20.001.pwBdGTJG.gz
```

Data feed files arrive in your bucket typically once an hour and each hour of usage is typically covered in a single data file. These files are compressed (gzip) before delivery into your bucket. We can write multiple files for a given hour of usage where files are very large (for example, when file contents for the hour exceed 50 MB before compression).

> **Note**
>
> If you don't have a Spot Instance running during a certain hour, you won't receive a data feed file for that hour.

The Spot Instance data feed files are tab-delimited. Each line in the data file corresponds to one instance-hour. Each line contains the fields listed in the following table.

Field	Description
Timestamp	The timestamp used to determine the price charged for this instance-hour.

Field	Description
UsageType	Indicates the type of usage and instance type being charged for. For m1.small Spot Instances, this field is set to "SpotUsage." For all other instance types, this field is set to "SpotUsage:{instance-type}," for example, "SpotUsage:c1.medium."
Operation	Indicates the product being charged for. For Linux/UNIX Spot Instances, this field is set to "RunInstances." For Microsoft Windows, this field is set to "RunInstances:0002." Spot usage is grouped according to Availability Zone.
InstanceID	The instance ID for the Spot Instance that generated this instance-hour.
MyBidID	The Spot Instance request ID for the request that generated this instance-hour.
MyMaxPrice	The maximum price specified for this Spot Instance request.
MarketPrice	The Spot Price at the time specified in the Timestamp field.
Charge	The price charged for this instance-hour.
Version	The version included in the data feed filename for this record.

Preparing Amazon S3 for Data Feeds

When you subscribe to data feeds, you tell Amazon EC2 which bucket you want to store the data feed file in. Before you subscribe to the data feed, consider the following when choosing your S3 bucket:

- You must have Amazon S3 FULL_CONTROL permission on the bucket you provide.
  If you're the bucket owner, you have this permission by default. If you're not the bucket owner, the bucket owner must grant your AWS account FULL_CONTROL permission.

- When you create your data feed subscription, Amazon EC2 updates the designated bucket's ACL to allow read and write permissions for the AWS data feeds account.

- Each data feed file has its own ACL (separate from the bucket's ACL).
  The bucket owner has FULL_CONTROL permission for the data files. The data feed account has read and write permission.

- Removing the permissions for the data feed account does not disable the data feed.
  If you remove those permissions but don't disable the data feed (which you do with the control API), we reinstate those permissions the next time the data feeds account needs to write a data file to your bucket.

- If you delete your data feed subscription, Amazon EC2 doesn't remove the read/write permissions for the data feed account on either the bucket or the data files.
  You must perform remove the read/write permissions yourself.

Subscribe to Your Spot Instance Data Feed

Command Line Tools

To subscribe to your Spot Instance data feed

- Enter the following command:

```
PROMPT>  ec2-create-spot-datafeed-subscription --bucket myawsbucket [--prefix
 prefix ]
```

Amazon EC2 returns output similar to the following:

```
SPOTDATAFEEDSUBSCRIPTION          111122223333    myawsbucket          prefix
Active
```

API

To subscribe to your Spot Instance data feed

- Construct the following Query request.

```
https://ec2.amazonaws.com/
?Action=CreateSpotDatafeedSubscription
&Bucket=myawsbucket
&Prefix=my-spot-subscription
&...auth parameters...
```

Following is an example response.

```
<CreateSpotDatafeedSubscriptionResponse xmlns="http://ec2.amazon
aws.com/doc/2012-07-20/">
  <requestId>59dbff89-35bd-4eac-99ed-be587EXAMPLE</requestId>
  <spotDatafeedSubscription>
        <ownerId>999988887777</ownerId>
        <bucket>myawsbucket</bucket>
        <prefix>my-spot-subscription</prefix>
        <state>Active</state>
        <fault>/</fault>
    </spotDatafeedSubscription>
</CreateSpotDatafeedSubscriptionResponse>
```

Delete a Spot Instance Data Feed

Command Line Tools

To delete a Spot Instance data feed

- To delete a data feed, enter the following command:

```
PROMPT>  ec2-delete-spot-datafeed-subscription
```

If the request is successful, the output is empty.

API

To delete a Spot Instance data feed

- Construct the following Query request.

```
https://ec2.amazonaws.com/
?Action=DeleteSpotDatafeedSubscription
&...auth parameters...
```

Following is an example response. It confirms that the subscription was deleted.

```
<DeleteSpotDatafeedSubscriptionResponse xmlns="http://ec2.amazon
aws.com/doc/2012-07-20/">
    <requestId>59dbff89-35bd-4eac-99ed-be587EXAMPLE</requestId>
    <return>true</return>
</DeleteSpotDatafeedSubscriptionResponse>
```

Programming Spot with AWS Java SDK

This section will hold the two SDK tutorials

You can AWS Java SDK to program Spot Instances:

- Tutorial: Amazon EC2 Spot Instances (p. 169)
- Tutorial: Advanced Amazon EC2 Spot Request Management (p. 178)

Tutorial: Amazon EC2 Spot Instances

Overview

Spot Instances allow you to bid on unused Amazon Elastic Compute Cloud (Amazon EC2) capacity and run the acquired instances for as long as your bid exceeds the current *Spot Price*. Amazon EC2 changes the Spot Price periodically based on supply and demand, and customers whose bids meet or exceed it gain access to the available Spot Instances. Like On-Demand Instances and Reserved Instances, Spot Instances provide you another option for obtaining more compute capacity.

Spot Instances can significantly lower your Amazon EC2 costs for batch processing, scientific research, image processing, video encoding, data and web crawling, financial analysis, and testing. Additionally, Spot Instances give you access to large amounts of additional capacity in situations where the need for that capacity is not urgent.

To use Spot Instances, place a Spot Instance request specifying the maximum price you are willing to pay per instance hour; this is your bid. If your bid exceeds the current Spot Price, your request is fulfilled and your instances will run until either you choose to terminate them or the Spot Price increases above your bid (whichever is sooner).

It's important to note two points:

- You will often pay less per hour than your bid. Amazon EC2 adjusts the Spot Price periodically as requests come in and available supply changes. Everyone pays the same Spot Price for that period regardless of whether their bid was higher. Therefore, you might pay less than your bid, but you will never pay more than your bid.
- If you're running Spot Instances and your bid no longer meets or exceeds the current Spot Price, your instances will be terminated. This means that you will want to make sure that your workloads and applications are flexible enough to take advantage of this opportunistic capacity.

Spot Instances perform exactly like other Amazon EC2 instances while running, and like other Amazon EC2 instances, Spot Instances can be terminated when you no longer need them. If you terminate your instance, you pay for any partial hour used (as you would for On-Demand or Reserved Instances). However, if the Spot Price goes above your bid and your instance is terminated by Amazon EC2, you will not be charged for any partial hour of usage.

This tutorial provides a quick overview of how to use the Java programming language to do the following.

- Submit a Spot Request
- Determine when the Spot Request becomes fulfilled
- Cancel the Spot Request
- Terminate associated instances

Prerequisites

To use this tutorial you need to be signed up for Amazon Web Services (AWS). If you have not yet signed up for AWS, go to the Amazon Web Services website, and click **Create an AWS Account** in the upper right corner of the page. In addition, you also need to install the AWS Java SDK.

If you are using the Eclipse development environment, we recommend that you install the AWS Toolkit for Eclipse. Note that the AWS Toolkit for Eclipse includes the latest version of the AWS SDK for Java.

Step 1: Setting Up Your Credentials

To begin using this code sample, you need to populate your credentials in the AwsCredentials.properties file. Specifically, you need to populate your secret key and access key.

To view your AWS access credentials

1. Go to the Amazon Web Services website at http://aws.amazon.com.

2. Click **My Account/Console**, and then click **Security Credentials**.

3. Under **Your Account**, click **Security Credentials**.

4. In the spaces provided, type your user name and password, and then click **Sign in using our secure server**.

5. Under **Access Credentials**, on the **Access Keys** tab, your access key ID is displayed. To view your secret key, under **Secret Access Key**, click **Show**.

Copy and paste your Access Key and Secret Access Key into the AwsCredentials.properties file. To learn more about setting up security credentials go to Amazon EC2 Security Credentials.

Now that you have configured your settings, you can get started using the code in the the example.

Step 2: Setting Up a Security Group

A *security group* acts as a firewall that controls the traffic allowed in and out of a group of instances. By default, an instance is started without any security group, which means that all incoming IP traffic, on any TCP port will be denied. So, before submitting our Spot Request, we will set up a security group that allows the necessary network traffic. For the purposes of this tutorial, we will create a new security group called "GettingStarted" that allows Secure Shell (SSH) traffic from the IP address where you are running your application from. To set up a new security group, you need to include or run the following code sample that sets up the security group programmatically.

After we create an `AmazonEC2` client object, we create a `CreateSecurityGroupRequest` object with the name, "GettingStarted" and a description for the security group. Then we call the `ec2.createSecurityGroup` API to create the group.

To enable access to the group, we create an `ipPermission` object with the IP address range set to the CIDR representation of the subnet for the local computer; the "/10" suffix on the IP address indicates the subnet for the specified IP address. We also configure the `ipPermission` object with the TCP protocol and port 22 (SSH). The final step is to call `ec2.authorizeSecurityGroupIngress` with the name of our security group and the `ipPermission` object.

```
 1
   // Retrieves the credentials from an AWSCredentials.properties file.
   AWSCredentials credentials = null;
   try {
 5     credentials = new PropertiesCredentials(
          GettingStartedApp.class.getResourceAsStream("AwsCredentials.proper
ties"));
   } catch (IOException e1) {
       System.out.println("Credentials were not properly entered into AwsCre
dentials.properties.");
       System.out.println(e1.getMessage());
10     System.exit(-1);
   }

   // Create the AmazonEC2Client object so we can call various APIs.
   AmazonEC2 ec2 = new AmazonEC2Client(credentials);
15
   // Create a new security group.
   try {
       CreateSecurityGroupRequest securityGroupRequest = new CreateSecurity
```

```
GroupRequest("GettingStartedGroup", "Getting Started Security Group");
       ec2.createSecurityGroup(securityGroupRequest);
20 } catch (AmazonServiceException ase) {
       // Likely this means that the group is already created, so ignore.
       System.out.println(ase.getMessage());
   }

25 String ipAddr = "0.0.0.0/0";

   // Get the IP of the current host, so that we can limit the Security
   // Group by default to the ip range associated with your subnet.
   try {
30     InetAddress addr = InetAddress.getLocalHost();

       // Get IP Address
       ipAddr = addr.getHostAddress()+"/10";
   } catch (UnknownHostException e) {
35 }

   // Create a range that you would like to populate.
   ArrayList<String> ipRanges = new ArrayList<String>();
   ipRanges.add(ipAddr);
40
   // Open up port 22 for TCP traffic to the associated IP
   // from above (e.g. ssh traffic).
   ArrayList<IpPermission> ipPermissions = new ArrayList<IpPermission> ();
   IpPermission ipPermission = new IpPermission();
45 ipPermission.setIpProtocol("tcp");
   ipPermission.setFromPort(new Integer(22));
   ipPermission.setToPort(new Integer(22));
   ipPermission.setIpRanges(ipRanges);
   ipPermissions.add(ipPermission);
50
   try {
       // Authorize the ports to the used.
       AuthorizeSecurityGroupIngressRequest ingressRequest =
          new AuthorizeSecurityGroupIngressRequest("GettingStartedGroup",ip
Permissions);
55     ec2.authorizeSecurityGroupIngress(ingressRequest);
   } catch (AmazonServiceException ase) {
       // Ignore because this likely means the zone has
       // already been authorized.
       System.out.println(ase.getMessage());
60 }
```

You can view this entire code sample in the `CreateSecurityGroupApp.java` code sample. Note you only need to run this application once to create a new security group.

You can also create the security group using the AWS Toolkit for Eclipse. Go to the toolkit documentation for more information.

Step 3: Submitting Your Spot Request

To submit a Spot request, you first need to determine the instance type, Amazon Machine Image (AMI), and maximum bid price you want to use. You must also include the security group we configured previously, so that you can log into the instance if desired.

There are several instance types to choose from; go to Amazon EC2 Instance Types for a complete list. For this tutorial, we will use t1.micro, the cheapest instance type available. Next, we will determine the type of AMI we would like to use. We'll use ami-8c1fece5, the most up-to-date Amazon Linux AMI available when we wrote this tutorial. The latest AMI may change over time, but you can always determine the latest version AMI by:

1. Logging into the AWS Management Console, clicking the EC2 tab, and, from the EC2 Console Dashboard, attempting to launch an instance.

2. In the window that displays AMIs, just use the AMI ID as shown in the following screen shot. Alternatively, you can use the `DescribeImages` API, but leveraging that command is outside the scope of this tutorial.

There are many ways to approach bidding for Spot instances; to get a broad overview of the various approaches you should view the Bidding for Spot Instances video. However, to get started, we'll describe three common strategies: bid to ensure cost is less than on-demand pricing; bid based on the value of the resulting computation; bid so as to acquire computing capacity as quickly as possible.

- **Reduce Cost below On-Demand** You have a batch processing job that will take a number of hours or days to run. However, you are flexible with respect to when it starts and when it completes. You want to see if you can complete it for less cost than with On-Demand Instances. You examine the Spot Price history for instance types using either the AWS Management Console or the Amazon EC2 API. For more information, go to Viewing Spot Price History. After you've analyzed the price history for your desired instance type in a given Availability Zone, you have two alternative approaches for your bid:
  - You could bid at the upper end of the range of Spot Prices (which are still below the On-Demand price), anticipating that your one-time Spot request would most likely be fulfilled and run for enough consecutive compute time to complete the job.
  - Or, you could bid at the lower end of the price range, and plan to combine many instances launched over time through a persistent request. The instances would run long enough--in aggregate--to complete the job at an even lower total cost. (We will explain how to automate this task later in this tutorial.)
- **Pay No More than the Value of the Result** You have a data processing job to run. You understand the value of the job's results well enough to know how much they are worth in terms of computing costs. After you've analyzed the Spot Price history for your instance type, you choose a bid price at which the cost of the computing time is no more than the value of the job's results. You create a persistent bid and allow it to run intermittently as the Spot Price fluctuates at or below your bid.
- **Acquire Computing Capacity Quickly** You have an unanticipated, short-term need for additional capacity that is not available through On-Demand Instances. After you've analyzed the Spot Price history for your instance type, you bid above the highest historical price to provide a high likelihood that your request will be fulfilled quickly and continue computing until it completes.

After you choose your bid price, you are ready to request a Spot Instance. For the purposes of this tutorial, we will bid the On-Demand price ($0.03) to maximize the chances that the bid will be fulfilled. You can determine the types of available instances and the On-Demand prices for instances by going to Amazon EC2 Pricing page. To request a Spot Instance, you simply need to build your request with the parameters you chose earlier. We start by creating a `RequestSpotInstanceRequest` object. The request object requires the number of instances you want to start and the bid price. Additionally, you need to set the `LaunchSpecification` for the request, which includes the instance type, AMI ID, and security group you want to use. Once the request is populated, you call the `requestSpotInstances` method on the `AmazonEC2Client` object. The following example shows how to request a Spot Instance.

```
  1
    // Retrieves the credentials from a AWSCrentials.properties file.
    AWSCredentials credentials = null;
    try {
  5     credentials = new PropertiesCredentials(
          GettingStartedApp.class.getResourceAsStream("AwsCredentials.proper
ties"));
    } catch (IOException e1) {
        System.out.println("Credentials were not properly entered into AwsCre
dentials.properties.");
        System.out.println(e1.getMessage());
 10     System.exit(-1);
    }

    // Create the AmazonEC2Client object so we can call various APIs.
    AmazonEC2 ec2 = new AmazonEC2Client(credentials);
 15
    // Initializes a Spot Instance Request
    RequestSpotInstancesRequest requestRequest = new RequestSpotInstances
Request();

    // Request 1 x t1.micro instance with a bid price of $0.03.
 20 requestRequest.setSpotPrice("0.03");
    requestRequest.setInstanceCount(Integer.valueOf(1));

    // Setup the specifications of the launch. This includes the
    // instance type (e.g. t1.micro) and the latest Amazon Linux
 25 // AMI id available. Note, you should always use the latest
    // Amazon Linux AMI id or another of your choosing.
    LaunchSpecification launchSpecification = new LaunchSpecification();
    launchSpecification.setImageId("ami-8c1fece5");
    launchSpecification.setInstanceType("t1.micro");
 30
    // Add the security group to the request.
    ArrayList<String> securityGroups = new ArrayList<String>();
    securityGroups.add("GettingStartedGroup");
    launchSpecification.setSecurityGroups(securityGroups);
 35
    // Add the launch specifications to the request.
    requestRequest.setLaunchSpecification(launchSpecification);

    // Call the RequestSpotInstance API.
 40 RequestSpotInstancesResult requestResult = ec2.requestSpotInstances(re
questRequest);
```

Running this code will launch a new Spot Instance Request. There are other options you can use to configure your Spot Requests. To learn more, please visit the Java Developers: Advanced Spot Features Tutorials or the RequestSpotInstances API in the Java SDK.

Note

You will be charged for any Spot Instances that are actually launched, so make sure that you cancel any requests and terminate any instances you launch to reduce any associated fees.

Step 4: Determining the State of Your Spot Request

Next, we want to create code to wait until the Spot request reaches the "active" state before proceeding to the last step. To determine the state of our Spot request, we poll the describeSpotInstanceRequests method for the state of the Spot request ID we want to monitor.

The request ID created in Step 2 is embedded in the response to our `requestSpotInstances` request. The following example code shows how to gather request IDs from the `requestSpotInstances` response and use them to populate an `ArrayList`.

```
 1
    // Call the RequestSpotInstance API.
    RequestSpotInstancesResult requestResult = ec2.requestSpotInstances(re
questRequest);
    List<SpotInstanceRequest> requestResponses = requestResult.getSpotInstance
Requests();
 5
    // Setup an arraylist to collect all of the request ids we want to
    // watch hit the running state.
    ArrayList<String> spotInstanceRequestIds = new ArrayList<String>();

10 // Add all of the request ids to the hashset, so we can determine when they
 hit the
    // active state.
    for (SpotInstanceRequest requestResponse : requestResponses) {
        System.out.println("Created Spot Request: "+requestResponse.getSpotIn
stanceRequestId());
        spotInstanceRequestIds.add(requestResponse.getSpotInstanceRequestId());
 15 }
```

To monitor your request ID, call the `describeSpotInstanceRequests` method to determine the state of the request. Then loop until the request is not in the "open" state. Note that we monitor for a state of not "open", rather a state of, say, "active", because the request can go straight to "closed" if there is a problem with your request arguments. The following code example provides the details of how to accomplish this task.

```
 1
    // Create a variable that will track whether there are any
    // requests still in the open state.
    boolean anyOpen;
 5
    do {
        // Create the describeRequest object with all of the request ids
        // to monitor (e.g. that we started).
        DescribeSpotInstanceRequestsRequest describeRequest = new DescribeSpot
InstanceRequestsRequest();
 10     describeRequest.setSpotInstanceRequestIds(spotInstanceRequestIds);

        // Initialize the anyOpen variable to false - which assumes there
        // are no requests open unless we find one that is still open.
        anyOpen=false;
 15
        try {
            // Retrieve all of the requests we want to monitor.
            DescribeSpotInstanceRequestsResult describeResult = ec2.describeS
potInstanceRequests(describeRequest);
```

```
          List<SpotInstanceRequest> describeResponses = describeResult.get
SpotInstanceRequests();
 20
          // Look through each request and determine if they are all in
          // the active state.
          for (SpotInstanceRequest describeResponse : describeResponses) {
              // If the state is open, it hasn't changed since we attempted
 25           // to request it. There is the potential for it to transition
              // almost immediately to closed or cancelled so we compare
              // against open instead of active.
          if (describeResponse.getState().equals("open")) {
              anyOpen = true;
 30           break;
          }
      }
    } catch (AmazonServiceException e) {
        // If we have an exception, ensure we don't break out of
 35     // the loop. This prevents the scenario where there was
        // blip on the wire.
        anyOpen = true;
    }

 40   try {
          // Sleep for 60 seconds.
          Thread.sleep(60*1000);
      } catch (Exception e) {
          // Do nothing because it woke up early.
 45   }
    } while (anyOpen);
```

After running this code, your Spot Instance Request will have completed or will have failed with an error that will be output to the screen. In either case, we can proceed to the next step to clean up any active requests and terminate any running instances.

Step 5: Cleaning Up Your Spot Requests and Instances

Lastly, we need to clean up our requests and instances. It is important to both cancel any outstanding requests *and* terminate any instances. Just canceling your requests will not terminate your instances, which means that you will continue to pay for them. If you terminate your instances, your Spot requests may be canceled, but there are some scenarios—such as if you use persistent bids—where terminating your instances is not sufficient to stop your request from being re-fulfilled. Therefore, it is a best practice to both cancel any active bids and terminate any running instances.

The following code demonstrates how to cancel your requests.

```
  1
   try {
       // Cancel requests.
       CancelSpotInstanceRequestsRequest cancelRequest = new CancelSpotInstan
ceRequestsRequest(spotInstanceRequestIds);
  5    ec2.cancelSpotInstanceRequests(cancelRequest);
   } catch (AmazonServiceException e) {
       // Write out any exceptions that may have occurred.
       System.out.println("Error cancelling instances");
       System.out.println("Caught Exception: " + e.getMessage());
 10    System.out.println("Reponse Status Code: " + e.getStatusCode());
```

```
        System.out.println("Error Code: " + e.getErrorCode());
        System.out.println("Request ID: " + e.getRequestId());
    }
```

To terminate any outstanding instances, you will need the instance ID associated with the request that started them. The following code example takes our original code for monitoring the instances and adds an `ArrayList` in which we store the instance ID associated with the `describeInstance` response.

```
 1
   // Create a variable that will track whether there are any requests
   // still in the open state.
   boolean anyOpen;
 5
   // Initialize variables.
   ArrayList<String> instanceIds = new ArrayList<String>();

   do {
10      // Create the describeRequest with all of the request ids to
        // monitor (e.g. that we started).
        DescribeSpotInstanceRequestsRequest describeRequest = new DescribeSpot
InstanceRequestsRequest();
        describeRequest.setSpotInstanceRequestIds(spotInstanceRequestIds);

15      // Initialize the anyOpen variable to false, which assumes there
        // are no requests open unless we find one that is still open.
        anyOpen = false;

        try {
20          // Retrieve all of the requests we want to monitor.
            DescribeSpotInstanceRequestsResult describeResult = ec2.describeS
potInstanceRequests(describeRequest);
            List<SpotInstanceRequest> describeResponses = describeResult.get
SpotInstanceRequests();

            // Look through each request and determine if they are all
25          // in the active state.
            for (SpotInstanceRequest describeResponse : describeResponses) {
                // If the state is open, it hasn't changed since we
                // attempted to request it. There is the potential for
                // it to transition almost immediately to closed or
30              // cancelled so we compare against open instead of active.
                if (describeResponse.getState().equals("open")) {
                    anyOpen = true;
                    break;
                }
35
            // Add the instance id to the list we will
            // eventually terminate.
            instanceIds.add(describeResponse.getInstanceId());
            }
40      } catch (AmazonServiceException e) {
            // If we have an exception, ensure we don't break out
            // of the loop. This prevents the scenario where there
            // was blip on the wire.
            anyOpen = true;
45      }
```

```
        try {
            // Sleep for 60 seconds.
            Thread.sleep(60*1000);
 50     } catch (Exception e) {
            // Do nothing because it woke up early.
        }
    } while (anyOpen);
```

Using the instance IDs, stored in the `ArrayList`, terminate any running instances using the following code snippet.

```
 1
   try {
        // Terminate instances.
        TerminateInstancesRequest terminateRequest = new TerminateInstances
Request(instanceIds);
 5      ec2.terminateInstances(terminateRequest);
    } catch (AmazonServiceException e) {
        // Write out any exceptions that may have occurred.
        System.out.println("Error terminating instances");
        System.out.println("Caught Exception: " + e.getMessage());
 10     System.out.println("Reponse Status Code: " + e.getStatusCode());
        System.out.println("Error Code: " + e.getErrorCode());
        System.out.println("Request ID: " + e.getRequestId());
    }
```

Bringing It All Together

To bring this all together, we provide a more object-oriented approach that combines the preceding steps we showed: initializing the EC2 Client, submitting the Spot Request, determining when the Spot Requests are no longer in the open state, and cleaning up any lingering Spot request and associated instances. We create a class called `Requests` that performs these actions.

We also create a `GettingStartedApp` class, which has a main method where we perform the high level function calls. Specifically, we initialize the `Requests` object described previously. We submit the Spot Instance request. Then we wait for the Spot request to reach the "Active" state. Finally, we clean up the requests and instances.

The complete source code is available for download at GitHub.

Congratulations! You have just completed the getting started tutorial for developing Spot Instance software with the AWS Java SDK.

Next Steps

We recommend that you take the Java Developers: Tutorial: Advanced Amazon EC2 Spot Request Management (p. 178).

Tutorial: Advanced Amazon EC2 Spot Request Management

Overview

Spot Instances allow you to bid on unused Amazon Elastic Compute Cloud (Amazon EC2) capacity and run those instances for as long as your bid exceeds the current Spot Price. Amazon EC2 changes the Spot Price periodically based on supply and demand. Customers whose bids meet or exceed the Spot Price gain access to the available Spot Instances. Like On-Demand Instances and Reserved Instances, Spot Instances provide you an additional option for obtaining more compute capacity.

Spot Instances can significantly lower your Amazon EC2 costs for batch processing, scientific research, image processing, video encoding, data and web crawling, financial analysis, and testing. Additionally, Spot Instances can provide access to large amounts of additional compute capacity when your need for the capacity is not urgent.

This tutorial provides a quick overview of some advanced Spot Request features, such as detailed options to create Spot requests, alternative methods for launching Spot Instances, and methods to manage your instances. This tutorial is not meant to be a complete list of all advanced topics associated with Spot Instances. Instead, it gives you a quick reference of code samples for some of the commonly used methods for managing Spot Requests and Spot Instances.

Prerequisites

To use this tutorial you need to be signed up for Amazon Web Services (AWS). If you have not yet signed up for AWS, go to the Amazon Web Services website, and click **Create an AWS Account** in the upper right corner of the page. In addition, you also need to install the AWS Java SDK.

If you are using the Eclipse development environment, we recommend that you install the AWS Toolkit for Eclipse. Note that the AWS Toolkit for Eclipse includes the latest version of the AWS SDK for Java.

Step 1: Setting Up Your Credentials

To begin using this code sample, you need to populate your credentials in the AwsCredentials.properties file. Specifically, you need to populate your secretKey and accessKey.

To view your AWS access credentials

1. Go to the Amazon Web Services website at http://aws.amazon.com.

2. Click **My Account/Console**, and then click **Security Credentials**.

3. Under **Your Account**, click **Security Credentials**.

4. In the spaces provided, type your user name and password, and then click **Sign in using our secure server**.

5. Under **Access Credentials**, on the **Access Keys** tab, your access key ID is displayed. To view your secret key, under **Secret Access Key**, click **Show**.

Copy and paste your Access Key and Secret Access Key into the AwsCredentials.properties file. To learn more about setting up security credentials go to Amazon EC2 Security Credentials.

Step 2: Setting Up a Security Group

Additionally, you need to configure your *security group*. A security group acts as a firewall that controls the traffic allowed in and out of a group of instances. By default, an instance is started without any security group, which means that all incoming IP traffic, on any TCP port will be denied. So, before submitting our Spot Request, we will set up a security group that allows the necessary network traffic. For the purposes of this tutorial, we will create a new security group called "GettingStarted" that allows Secure Shell (SSH)

traffic from the IP address where you are running your application from. To set up a new security group, you need to include or run the following code sample that sets up the security group programmatically.

After we create an `AmazonEC2` client object, we create a `CreateSecurityGroupRequest` object with the name, "GettingStarted" and a description for the security group. Then we call the `ec2.createSecurityGroup` API to create the group.

To enable access to the group, we create an `ipPermission` object with the IP address range set to the CIDR representation of the subnet for the local computer; the "/10" suffix on the IP address indicates the subnet for the specified IP address. We also configure the `ipPermission` object with the TCP protocol and port 22 (SSH). The final step is to call `ec2.authorizeSecurityGroupIngress` with the name of our security group and the `ipPermission` object.

(The following code is the same as what we used in the first tutorial.)

```
 1
    // Retrieves the credentials from an AWSCredentials.properties file.
    AWSCredentials credentials = null;
    try {
 5      credentials = new PropertiesCredentials(
          GettingStartedApp.class.getResourceAsStream("AwsCredentials.proper
ties"));
    } catch (IOException e1) {
      System.out.println("Credentials were not properly entered into AwsCreden
tials.properties.");
        System.out.println(e1.getMessage());
 10     System.exit(-1);
    }

    // Create the AmazonEC2Client object so we can call various APIs.
    AmazonEC2 ec2 = new AmazonEC2Client(credentials);
 15
    // Create a new security group.
    try {
        CreateSecurityGroupRequest securityGroupRequest =
            new CreateSecurityGroupRequest("GettingStartedGroup", "Getting
Started Security Group");
 20     ec2.createSecurityGroup(securityGroupRequest);
    } catch (AmazonServiceException ase) {
        // Likely this means that the group is already created, so ignore.
        System.out.println(ase.getMessage());
    }
 25
    String ipAddr = "0.0.0.0/0";

    // Get the IP of the current host, so that we can limit the Security Group
    // by default to the ip range associated with your subnet.
 30 try {
        InetAddress addr = InetAddress.getLocalHost();

        // Get IP Address
        ipAddr = addr.getHostAddress()+"/10";
 35 } catch (UnknownHostException e) {
    }

    // Create a range that you would like to populate.
    ArrayList<String> ipRanges = new ArrayList<String>();
 40 ipRanges.add(ipAddr);
```

```
      // Open up port 22 for TCP traffic to the associated IP from
      // above (e.g. ssh traffic).
      ArrayList<IpPermission> ipPermissions = new ArrayList<IpPermission> ();
 45   IpPermission ipPermission = new IpPermission();
      ipPermission.setIpProtocol("tcp");
      ipPermission.setFromPort(new Integer(22));
      ipPermission.setToPort(new Integer(22));
      ipPermission.setIpRanges(ipRanges);
 50   ipPermissions.add(ipPermission);

      try {
          // Authorize the ports to the used.
          AuthorizeSecurityGroupIngressRequest ingressRequest =
 55          new AuthorizeSecurityGroupIngressRequest("GettingStartedGroup",ip
Permissions);
          ec2.authorizeSecurityGroupIngress(ingressRequest);
      } catch (AmazonServiceException ase) {
          // Ignore because this likely means the zone has already
          // been authorized.
 60      System.out.println(ase.getMessage());
      }
```

You can view this entire code sample in the `advanced.CreateSecurityGroupApp.java` code sample. Note you only need to run this application once to create a new security group.

You can also create the security group using the AWS Toolkit for Eclipse. Go to the toolkit documentation for more information.

Detailed Spot Instance Request Creation Options

As we explained in Tutorial: Amazon EC2 Spot Instances (p. 169), you need to build your request with an instance type, an Amazon Machine Image (AMI), and maximum bid price.

Let's start by creating a `RequestSpotInstanceRequest` object. The request object requires the number of instances you want and the bid price. Additionally, we need to set the `LaunchSpecification` for the request, which includes the instance type, AMI ID, and security group you want to use. After the request is populated, we call the `requestSpotInstances` method on the `AmazonEC2Client` object. An example of how to request a Spot instance follows.

(The following code is the same as what we used in the first tutorial.)

```
  1
    // Retrieves the credentials from an AWSCredentials.properties file.
    AWSCredentials credentials = null;
    try {
  5    credentials = new PropertiesCredentials(
          GettingStartedApp.class.getResourceAsStream("AwsCredentials.proper
ties"));
    } catch (IOException e1) {
        System.out.println("Credentials were not properly entered into AwsCre
dentials.properties.");
        System.out.println(e1.getMessage());
 10     System.exit(-1);
    }
```

```
    // Create the AmazonEC2Client object so we can call various APIs.
    AmazonEC2 ec2 = new AmazonEC2Client(credentials);
 15
    // Initializes a Spot Instance Request
    RequestSpotInstancesRequest requestRequest = new RequestSpotInstances
Request();

    // Request 1 x t1.micro instance with a bid price of $0.03.
 20 requestRequest.setSpotPrice("0.03");
    requestRequest.setInstanceCount(Integer.valueOf(1));

    // Set up the specifications of the launch. This includes the
    // instance type (e.g. t1.micro) and the latest Amazon Linux
 25 // AMI id available. Note, you should always use the latest
    // Amazon Linux AMI id or another of your choosing.
    LaunchSpecification launchSpecification = new LaunchSpecification();
    launchSpecification.setImageId("ami-8c1fece5");
    launchSpecification.setInstanceType("t1.micro");
 30
    // Add the security group to the request.
    ArrayList<String> securityGroups = new ArrayList<String>();
    securityGroups.add("GettingStartedGroup");
    launchSpecification.setSecurityGroups(securityGroups);
 35
    // Add the launch specification.
    requestRequest.setLaunchSpecification(launchSpecification);

    // Call the RequestSpotInstance API.
 40 RequestSpotInstancesResult requestResult = ec2.requestSpotInstances(re
questRequest);
```

Persistent vs. One-Time Requests

When building a Spot request, you can specify several optional parameters. The first is whether your request is one-time only or persistent. By default, it is a one-time request. A one-time request can be fulfilled only once, and after the requested instances are terminated, the request will be closed. A persistent request is considered for fulfillment whenever there is no Spot Instance running for the same request. To specify the type of request, you simply need to set the Type on the Spot request. This can be done with the following code.

```
  1
    // Retrieves the credentials from an
    // AWSCredentials.properties file.
    AWSCredentials credentials = null;
  5 try {
        credentials = new PropertiesCredentials(
            GettingStartedApp.class.getResourceAsStream("AwsCredentials.proper
ties"));
    } catch (IOException e1) {
        System.out.println("Credentials were not properly entered into AwsCre
dentials.properties.");
 10     System.out.println(e1.getMessage());
        System.exit(-1);
    }

    // Create the AmazonEC2Client object so we can call various APIs.
```

```
15 AmazonEC2 ec2 = new AmazonEC2Client(credentials);

   // Initializes a Spot Instance Request
   RequestSpotInstancesRequest requestRequest = new RequestSpotInstances
Request();

20 // Request 1 x t1.micro instance with a bid price of $0.03.
   requestRequest.setSpotPrice("0.03");
   requestRequest.setInstanceCount(Integer.valueOf(1));

   // Set the type of the bid to persistent.
25 requestRequest.setType("persistent");

   // Set up the specifications of the launch. This includes the
   // instance type (e.g. t1.micro) and the latest Amazon Linux
   // AMI id available. Note, you should always use the latest
30 // Amazon Linux AMI id or another of your choosing.
   LaunchSpecification launchSpecification = new LaunchSpecification();
   launchSpecification.setImageId("ami-8c1fece5");
   launchSpecification.setInstanceType("t1.micro");

35 // Add the security group to the request.
   ArrayList<String> securityGroups = new ArrayList<String>();
   securityGroups.add("GettingStartedGroup");
   launchSpecification.setSecurityGroups(securityGroups);

40 // Add the launch specification.
   requestRequest.setLaunchSpecification(launchSpecification);

   // Call the RequestSpotInstance API.
   RequestSpotInstancesResult requestResult = ec2.requestSpotInstances(re
questRequest);
45
```

Limiting the Duration of a Request

You can also optionally specify the length of time that your request will remain valid. You can specify both a starting and ending time for this period. By default, a Spot request will be considered for fulfillment from the moment it is created until it is either fulfilled or canceled by you. However you can constrain the validity period if you need to. An example of how to specify this period is shown in the following code.

```
 1
   // Retrieves the credentials from an AWSCredentials.properties file.
   AWSCredentials credentials = null;
   try {
 5     credentials = new PropertiesCredentials(
         GettingStartedApp.class.getResourceAsStream("AwsCredentials.proper
ties"));
   } catch (IOException e1) {
       System.out.println("Credentials were not properly entered into AwsCre
dentials.properties.");
       System.out.println(e1.getMessage());
10     System.exit(-1);
   }

   // Create the AmazonEC2Client object so we can call various APIs.
   AmazonEC2 ec2 = new AmazonEC2Client(credentials);
```

```
15
   // Initializes a Spot Instance Request
   RequestSpotInstancesRequest requestRequest = new RequestSpotInstances
Request();

   // Request 1 x t1.micro instance with a bid price of $0.03.
20 requestRequest.setSpotPrice("0.03");
   requestRequest.setInstanceCount(Integer.valueOf(1));

   // Set the valid start time to be two minutes from now.
   Calendar cal = Calendar.getInstance();
25 cal.add(Calendar.MINUTE, 2);
   requestRequest.setValidFrom(cal.getTime());

   // Set the valid end time to be two minutes and two hours from now.
   cal.add(Calendar.HOUR, 2);
30 requestRequest.setValidUntil(cal.getTime());

   // Set up the specifications of the launch. This includes
   // the instance type (e.g. t1.micro)

35 // and the latest Amazon Linux AMI id available.
   // Note, you should always use the latest Amazon
   // Linux AMI id or another of your choosing.
   LaunchSpecification launchSpecification = new LaunchSpecification();
   launchSpecification.setImageId("ami-8c1fece5");
40 launchSpecification.setInstanceType("t1.micro");

   // Add the security group to the request.
   ArrayList<String> securityGroups = new ArrayList<String>();
   securityGroups.add("GettingStartedGroup");
45 launchSpecification.setSecurityGroups(securityGroups);

   // Add the launch specification.
   requestRequest.setLaunchSpecification(launchSpecification);

50 // Call the RequestSpotInstance API.
   RequestSpotInstancesResult requestResult = ec2.requestSpotInstances(re
questRequest);
```

Grouping Your Amazon EC2 Spot Instance Requests

You have the option of grouping your Spot instance requests in several different ways. We'll look at the benefits of using launch groups, Availability Zone groups, and placement groups.

If you want to ensure your Spot instances are all launched and terminated together, then you have the option to leverage a launch group. A launch group is a label that groups a set of bids together. All instances in a launch group are started and terminated together. Note, if instances in a launch group have already been fulfilled, there is no guarantee that new instances launched with the same launch group will also be fulfilled. An example of how to set a Launch Group is shown in the following code example.

```
1
   // Retrieves the credentials from an AWSCredentials.properties file.
   AWSCredentials credentials = null;
   try {
5     credentials = new PropertiesCredentials(
```

```
           GettingStartedApp.class.getResourceAsStream("AwsCredentials.proper
ties"));
    } catch (IOException e1) {
        System.out.println("Credentials were not properly entered into AwsCre
dentials.properties.");
        System.out.println(e1.getMessage());
 10     System.exit(-1);
    }

    // Create the AmazonEC2Client object so we can call various APIs.
    AmazonEC2 ec2 = new AmazonEC2Client(credentials);
 15
    // Initializes a Spot Instance Request
    RequestSpotInstancesRequest requestRequest = new RequestSpotInstances
Request();

    // Request 5 x t1.micro instance with a bid price of $0.03.
 20 requestRequest.setSpotPrice("0.03");
    requestRequest.setInstanceCount(Integer.valueOf(5));

    // Set the launch group.
    requestRequest.setLaunchGroup("ADVANCED-DEMO-LAUNCH-GROUP");
 25
    // Set up the specifications of the launch. This includes
    // the instance type (e.g. t1.micro) and the latest Amazon Linux
    // AMI id available. Note, you should always use the latest
    // Amazon Linux AMI id or another of your choosing.
 30 LaunchSpecification launchSpecification = new LaunchSpecification();
    launchSpecification.setImageId("ami-8c1fece5");
    launchSpecification.setInstanceType("t1.micro");

    // Add the security group to the request.
 35 ArrayList<String> securityGroups = new ArrayList<String>();
    securityGroups.add("GettingStartedGroup");
    launchSpecification.setSecurityGroups(securityGroups);

    // Add the launch specification.
 40 requestRequest.setLaunchSpecification(launchSpecification);

    // Call the RequestSpotInstance API.
    RequestSpotInstancesResult requestResult = ec2.requestSpotInstances(re
questRequest);
```

If you want to ensure that all instances within a request are launched in the same Availability Zone, and you don't care which one, you can leverage Availability Zone groups. An Availability Zone group is a label that groups a set of instances together in the same Availability Zone. All instances that share an Availability Zone group and are fulfilled at the same time will start in the same Availability Zone. An example of how to set an Availability Zone group follows.

```
  1
    // Retrieves the credentials from an AWSCredentials.properties file.
    AWSCredentials credentials = null;
    try {
  5    credentials = new PropertiesCredentials(
           GettingStartedApp.class.getResourceAsStream("AwsCredentials.proper
ties"));
```

```
    } catch (IOException e1) {
        System.out.println("Credentials were not properly entered into AwsCre
dentials.properties.");
        System.out.println(e1.getMessage());
 10     System.exit(-1);
    }

    // Create the AmazonEC2Client object so we can call various APIs.
    AmazonEC2 ec2 = new AmazonEC2Client(credentials);
 15
    // Initializes a Spot Instance Request
    RequestSpotInstancesRequest requestRequest = new RequestSpotInstances
Request();

    // Request 5 x t1.micro instance with a bid price of $0.03.
 20 requestRequest.setSpotPrice("0.03");
    requestRequest.setInstanceCount(Integer.valueOf(5));

    // Set the availability zone group.
    requestRequest.setAvailabilityZoneGroup("ADVANCED-DEMO-AZ-GROUP");
 25
    // Set up the specifications of the launch.  This includes the instance
    // type (e.g.  t1.micro) and the latest Amazon Linux AMI id available.
    // Note, you should always use the latest Amazon Linux AMI id or another
    // of your choosing.
 30 LaunchSpecification launchSpecification = new LaunchSpecification();
    launchSpecification.setImageId("ami-8c1fece5");
    launchSpecification.setInstanceType("t1.micro");

    // Add the security group to the request.
 35 ArrayList<String> securityGroups = new ArrayList<String>();
    securityGroups.add("GettingStartedGroup");
    launchSpecification.setSecurityGroups(securityGroups);

    // Add the launch specification.
 40 requestRequest.setLaunchSpecification(launchSpecification);

    // Call the RequestSpotInstance API.
    RequestSpotInstancesResult requestResult = ec2.requestSpotInstances(re
questRequest);
```

You can specify an Availability Zone that you want for your Spot Instances. The following code example shows you how to set an Availability Zone.

```
  1
    // Retrieves the credentials from an AWSCredentials.properties file.
    AWSCredentials credentials = null;
    try {
  5     credentials = new PropertiesCredentials(
            GettingStartedApp.class.getResourceAsStream("AwsCredentials.proper
ties"));
    } catch (IOException e1) {
        System.out.println("Credentials were not properly entered into AwsCre
dentials.properties.");
        System.out.println(e1.getMessage());
 10     System.exit(-1);
```

```
      }

      // Create the AmazonEC2Client object so we can call various APIs.
      AmazonEC2 ec2 = new AmazonEC2Client(credentials);
 15
      // Initializes a Spot Instance Request
      RequestSpotInstancesRequest requestRequest = new RequestSpotInstances
 Request();

      // Request 1 x t1.micro instance with a bid price of $0.03.
 20   requestRequest.setSpotPrice("0.03");
      requestRequest.setInstanceCount(Integer.valueOf(1));

      // Set up the specifications of the launch.  This includes the instance
      // type (e.g.  t1.micro) and the latest Amazon Linux AMI id available.
 25   // Note, you should always use the latest Amazon Linux AMI id or another
      // of your choosing.
      LaunchSpecification launchSpecification = new LaunchSpecification();
      launchSpecification.setImageId("ami-8c1fece5");
      launchSpecification.setInstanceType("t1.micro");
 30
      // Add the security group to the request.
      ArrayList<String> securityGroups = new ArrayList<String>();
      securityGroups.add("GettingStartedGroup");
      launchSpecification.setSecurityGroups(securityGroups);
 35
      // Set up the availability zone to use.  Note we could retrieve the
      // availability zones using the ec2.describeAvailabilityZones() API.  For
      // this demo we will just use us-east-1a.
      SpotPlacement placement = new SpotPlacement("us-east-1b");
 40
      launchSpecification.setPlacement(placement);

      // Add the launch specification.
      requestRequest.setLaunchSpecification(launchSpecification);
 45
      // Call the RequestSpotInstance API.
      RequestSpotInstancesResult requestResult = ec2.requestSpotInstances(re
 questRequest);
```

Lastly, you can specify a *placement group* if you are using High Performance Computing (HPC) Spot instances, such as cluster compute instances or cluster GPU instances. Placement groups provide you with lower latency and high-bandwidth connectivity between the instances. An example of how to set a placement group follows.

```
  1
      // Retrieves the credentials from an AWSCredentials.properties file.
      AWSCredentials credentials = null;
      try {
  5      credentials = new PropertiesCredentials(
             GettingStartedApp.class.getResourceAsStream("AwsCredentials.proper
 ties"));
      } catch (IOException e1) {
         System.out.println("Credentials were not properly entered into AwsCre
 dentials.properties.");
         System.out.println(e1.getMessage());
```

```
 10     System.exit(-1);
    }

    // Create the AmazonEC2Client object so we can call various APIs.
    AmazonEC2 ec2 = new AmazonEC2Client(credentials);
 15
    // Initializes a Spot Instance Request
    RequestSpotInstancesRequest requestRequest = new RequestSpotInstances
Request();

    // Request 1 x t1.micro instance with a bid price of $0.03.
 20 requestRequest.setSpotPrice("0.03");
    requestRequest.setInstanceCount(Integer.valueOf(1));

    // Set up the specifications of the launch. This includes the instance
    // type (e.g.  t1.micro) and the latest Amazon Linux AMI id available.
 25 // Note, you should always use the latest Amazon Linux AMI id or another
    // of your choosing.
    LaunchSpecification launchSpecification = new LaunchSpecification();
    launchSpecification.setImageId("ami-8c1fece5");
    launchSpecification.setInstanceType("t1.micro");
 30
    // Add the security group to the request.
    ArrayList<String> securityGroups = new ArrayList<String>();
    securityGroups.add("GettingStartedGroup");
    launchSpecification.setSecurityGroups(securityGroups);
 35
    // Set up the placement group to use with whatever name you desire.
    // For this demo we will just use "ADVANCED-DEMO-PLACEMENT-GROUP".
    SpotPlacement placement = new SpotPlacement();
    placement.setGroupName("ADVANCED-DEMO-PLACEMENT-GROUP");
 40 launchSpecification.setPlacement(placement);

    // Add the launch specification.
    requestRequest.setLaunchSpecification(launchSpecification);

 45 // Call the RequestSpotInstance API.
    RequestSpotInstancesResult requestResult = ec2.requestSpotInstances(re
questRequest);
```

All of the parameters shown in this section are optional. It is also important to realize that most of these parameters—with the exception of whether your bid is one-time or persistent—can reduce the likelihood of bid fulfillment. So, it is important to leverage these options only if you need them. All of the preceding code examples are combined into one long code sample, which can be found in the `com.amazonaws.codesamples.advanced.InlineGettingStartedCodeSampleApp.java` class.

How to Persist a Root Partition After Interruption or Termination

One of the easiest ways to manage interruption of your Spot instances is to ensure that your data is checkpointed to an Amazon Elastic Block Store (Amazon EBS) volume on a regular cadence. By checkpointing periodically, if there is an interruption you will lose only the data created since the last checkpoint (assuming no other non-idempotent actions are performed in between). To make this process easier, you can configure your Spot Request to ensure that your root partition will not be deleted on interruption or termination. We've inserted new code in the following example that shows how to enable this scenario.

In the added code, we create a `BlockDeviceMapping` object and set its associated Elastic Block Storage (EBS) to an EBS object that we've configured to `not` be deleted if the Spot Instance is terminated. We then add this `BlockDeviceMapping` to the ArrayList of mappings that we include in the launch specification.

```
 1
    // Retrieves the credentials from an AWSCredentials.properties file.
    AWSCredentials credentials = null;
    try {
 5      credentials = new PropertiesCredentials(
            GettingStartedApp.class.getResourceAsStream("AwsCredentials.proper
ties"));
    } catch (IOException e1) {
        System.out.println("Credentials were not properly entered into AwsCre
dentials.properties.");
        System.out.println(e1.getMessage());
 10     System.exit(-1);
    }

    // Create the AmazonEC2Client object so we can call various APIs.
    AmazonEC2 ec2 = new AmazonEC2Client(credentials);
 15
    // Initializes a Spot Instance Request
    RequestSpotInstancesRequest requestRequest = new RequestSpotInstances
Request();

    // Request 1 x t1.micro instance with a bid price of $0.03.
 20 requestRequest.setSpotPrice("0.03");
    requestRequest.setInstanceCount(Integer.valueOf(1));

    // Set up the specifications of the launch.  This includes the instance
    // type (e.g.  t1.micro) and the latest Amazon Linux AMI id available.
 25 // Note, you should always use the latest Amazon Linux AMI id or another
    // of your choosing.
    LaunchSpecification launchSpecification = new LaunchSpecification();
    launchSpecification.setImageId("ami-8c1fece5");
    launchSpecification.setInstanceType("t1.micro");
 30
    // Add the security group to the request.
    ArrayList<String> securityGroups = new ArrayList<String>();
    securityGroups.add("GettingStartedGroup");
    launchSpecification.setSecurityGroups(securityGroups);
 35
    // Create the block device mapping to describe the root partition.
    BlockDeviceMapping blockDeviceMapping = new BlockDeviceMapping();
    blockDeviceMapping.setDeviceName("/dev/sda1");

 40 // Set the delete on termination flag to false.
    EbsBlockDevice ebs = new EbsBlockDevice();
    ebs.setDeleteOnTermination(Boolean.FALSE);
    blockDeviceMapping.setEbs(ebs);

 45 // Add the block device mapping to the block list.
    ArrayList<BlockDeviceMapping> blockList = new ArrayList<BlockDeviceMap
ping>();
    blockList.add(blockDeviceMapping);

    // Set the block device mapping configuration in the launch specifications.
```

```
50  launchSpecification.setBlockDeviceMappings(blockList);

    // Add the launch specification.
    requestRequest.setLaunchSpecification(launchSpecification);

55  // Call the RequestSpotInstance API.
    RequestSpotInstancesResult requestResult = ec2.requestSpotInstances(re
questRequest);
```

Assuming you wanted to re-attach this volume to your instance on startup, you can also use the block device mapping settings. Alternatively, if you attached a non-root partition, you can specify the Amazon EBS volumes you want to attach to your Spot instance after it resumes. You do this simply by specifying a snapshot ID in your `EbsBlockDevice` and alternative device name in your `BlockDeviceMapping` objects. By leveraging block device mappings, it can be easier to bootstrap your instance.

Using the root partition to checkpoint your critical data is a great way to manage the potential for interruption of your instances. For more methods on managing the potential of interruption, please visit the Managing Interruption video.

How to Tag Your Spot Requests and Instances

Adding tags to EC2 resources can simplify the administration of your cloud infrastructure. A form of metadata, tags can be used to create user-friendly names, enhance searchability, and improve coordination between multiple users. You can also use tags to automate scripts and portions of your processes.

To add tags to your resources, you need to tag them *after* they have been requested. Specifically, you must add a tag after a Spot request has been submitted or after the `RunInstances` call has been performed. The following code example illustrates adding tags.

```
 1
   /*
    * Copyright 2010-2011 Amazon.com, Inc. or its affiliates. All Rights Re
served.
    *
 5  * Licensed under the Apache License, Version 2.0 (the "License").
    * You may not use this file except in compliance with the License.
    * A copy of the License is located at
    *
    *   http://aws.amazon.com/apache2.0
10  *
    * or in the "license" file accompanying this file. This file is distributed
    * on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
    * express or implied. See the License for the specific language governing
    * permissions and limitations under the License.
15  */
    package com.amazonaws.codesamples.advanced;

    import java.io.IOException;
    import java.util.ArrayList;
20  import java.util.List;

    import com.amazonaws.AmazonServiceException;
    import com.amazonaws.auth.AWSCredentials;
    import com.amazonaws.auth.PropertiesCredentials;
25  import com.amazonaws.codesamples.getting_started.GettingStartedApp;
    import com.amazonaws.services.ec2.AmazonEC2;
```

```
   import com.amazonaws.services.ec2.AmazonEC2Client;
   import com.amazonaws.services.ec2.model.CancelSpotInstanceRequestsRequest;
   import com.amazonaws.services.ec2.model.CreateTagsRequest;
30 import com.amazonaws.services.ec2.model.DescribeSpotInstanceRequestsRequest;
   import com.amazonaws.services.ec2.model.DescribeSpotInstanceRequestsResult;
   import com.amazonaws.services.ec2.model.LaunchSpecification;
   import com.amazonaws.services.ec2.model.RequestSpotInstancesRequest;
   import com.amazonaws.services.ec2.model.RequestSpotInstancesResult;
35 import com.amazonaws.services.ec2.model.SpotInstanceRequest;
   import com.amazonaws.services.ec2.model.Tag;
   import com.amazonaws.services.ec2.model.TerminateInstancesRequest;


   /**
40  * Welcome to your new AWS Java SDK based project!
    *
    * This class is meant as a starting point for your console-based application
 that
    * makes one or more calls to the AWS services supported by the Java SDK,
such as EC2,
    * SimpleDB, and S3.
45  *
    * In order to use the services in this sample, you need:
    *
    *  - A valid Amazon Web Services account. You can register for AWS at:
    *       https://aws-portal.amazon.com/gp/aws/developer/registration/in
dex.html
50  *
    *  - Your account's Access Key ID and Secret Access Key:
    *       http://aws.amazon.com/security-credentials
    *
    *  - A subscription to Amazon EC2. You can sign up for EC2 at:
55  *       http://aws.amazon.com/ec2/
    *
    */

   public class InlineTaggingCodeSampleApp {
60
     /**
      * @param args
      */
     public static void main(String[] args) {
65     //============================================================//
       //=============== Submitting a Request =======================//
       //============================================================//

       // Retrieves the credentials from an AWSCredentials.properties file.
70     AWSCredentials credentials = null;
       try {
         credentials = new PropertiesCredentials(
          GettingStartedApp.class.getResourceAsStream("AwsCredentials.proper
ties"));
       } catch (IOException e1) {
75     System.out.println("Credentials were not properly entered into
AwsCredentials.properties.");
         System.out.println(e1.getMessage());
         System.exit(-1);
       }
```

```
 80     // Create the AmazonEC2Client object so we can
        // call various APIs.
        AmazonEC2 ec2 = new AmazonEC2Client(credentials);


        // Initializes a Spot Instance Request
 85     RequestSpotInstancesRequest requestRequest = new RequestSpotInstances
Request();


        // Request 1 x t1.micro instance with a bid price of $0.03.
        requestRequest.setSpotPrice("0.03");
        requestRequest.setInstanceCount(Integer.valueOf(1));
 90
        // Set up the specifications of the launch. This includes
        // the instance type (e.g. t1.micro) and the latest Amazon
        // Linux AMI id available. Note, you should always use the
        // latest Amazon Linux AMI id or another of your choosing.
 95     LaunchSpecification launchSpecification = new LaunchSpecification();
        launchSpecification.setImageId("ami-8c1fece5");
        launchSpecification.setInstanceType("t1.micro");


        // Add the security group to the request.
100     ArrayList<String> securityGroups = new ArrayList<String>();
        securityGroups.add("GettingStartedGroup");
        launchSpecification.setSecurityGroups(securityGroups);


        // Add the launch specifications to the request.
105     requestRequest.setLaunchSpecification(launchSpecification);


        //===========================================================//
        //======= Getting the Request ID from the Request ==========//
        //===========================================================//
110
        // Call the RequestSpotInstance API.
       RequestSpotInstancesResult requestResult = ec2.requestSpotInstances(re
questRequest);
        List<SpotInstanceRequest> requestResponses = requestResult.getSpotIn
stanceRequests();


115     // Set up an arraylist to collect all of the request ids we want to
        // watch hit the running state.
        ArrayList<String> spotInstanceRequestIds = new ArrayList<String>();


        // Add all of the request ids to the hashset, so we can
120     // determine when they hit the active state.
        for (SpotInstanceRequest requestResponse : requestResponses) {
          System.out.println("Created Spot Request: "+requestResponse.getSpot
InstanceRequestId());
         spotInstanceRequestIds.add(requestResponse.getSpotInstanceRequestId());
        }
125
        //=========================================================//
        //============= Tag the Spot Requests =====================//
        //=========================================================//


130     // Create the list of tags we want to create
        ArrayList<Tag> requestTags = new ArrayList<Tag>();
        requestTags.add(new Tag("keyname1","value1"));
```

```
        // Create a tag request for the requests.
135     CreateTagsRequest createTagsRequest_requests = new CreateTagsRequest();
        createTagsRequest_requests.setResources(spotInstanceRequestIds);
        createTagsRequest_requests.setTags(requestTags);


        // Try to tag the Spot request submitted.
140     try {
          ec2.createTags(createTagsRequest_requests);
        } catch (AmazonServiceException e) {
          // Write out any exceptions that may have occurred.
          System.out.println("Error terminating instances");
145       System.out.println("Caught Exception: " + e.getMessage());
          System.out.println("Reponse Status Code: " + e.getStatusCode());
          System.out.println("Error Code: " + e.getErrorCode());
          System.out.println("Request ID: " + e.getRequestId());
        }
150
        //===========================================================//
        //====== Determining the State of the Spot Request =========//
        //===========================================================//

155     // Create a variable that will track whether there are any
        // requests still in the open state.
        boolean anyOpen;

        // Initialize variables.
160     ArrayList<String> instanceIds = new ArrayList<String>();

        do {
          // Create the describeRequest with tall of the request
          // id to monitor (e.g. that we started).
165       DescribeSpotInstanceRequestsRequest describeRequest = new DescribeS
potInstanceRequestsRequest();
          describeRequest.setSpotInstanceRequestIds(spotInstanceRequestIds);

          // Initialize the anyOpen variable to false - which assumes there are
 no requests open unless
          // we find one that is still open.
170       anyOpen = false;

          try {
            // Retrieve all of the requests we want to monitor.
            DescribeSpotInstanceRequestsResult describeResult = ec2.describeS
potInstanceRequests(describeRequest);
175         List<SpotInstanceRequest> describeResponses = describeResult.get
SpotInstanceRequests();

            // Look through each request and determine if they are all
            // in the active state.
            for (SpotInstanceRequest describeResponse : describeResponses) {
180           // If the state is open, it hasn't changed since we
              // attempted to request it. There is the potential
              // for it to transition almost immediately to closed or
              // canceled so we compare against open instead of active.
              if (describeResponse.getState().equals("open")) {
185             anyOpen = true;
                break;
              }
```

```
              // Add the instance id to the list we will
190           // eventually terminate.
              instanceIds.add(describeResponse.getInstanceId());
            }
          } catch (AmazonServiceException e) {
            // If we have an exception, ensure we don't break out
195         // of the loop. This prevents the scenario where there
            // was blip on the wire.
            anyOpen = true;
          }

200       try {
            // Sleep for 60 seconds.
            Thread.sleep(60*1000);
          } catch (Exception e) {
            // Do nothing because it woke up early.
205       }
        } while (anyOpen);


        //===========================================================//
        //============= Tag the Spot Instances ====================//
210     //===========================================================//


        // Create the list of tags we want to create
        ArrayList<Tag> instanceTags = new ArrayList<Tag>();
        instanceTags.add(new Tag("keyname1","value1"));
215
        // Create a tag request for instances.
        CreateTagsRequest createTagsRequest_instances = new CreateTagsRequest();
        createTagsRequest_instances.setResources(instanceIds);
        createTagsRequest_instances.setTags(instanceTags);
220
        // Try to tag the Spot instance started.
        try {
          ec2.createTags(createTagsRequest_instances);
        } catch (AmazonServiceException e) {
225       // Write out any exceptions that may have occurred.
          System.out.println("Error terminating instances");
          System.out.println("Caught Exception: " + e.getMessage());
          System.out.println("Reponse Status Code: " + e.getStatusCode());
          System.out.println("Error Code: " + e.getErrorCode());
230       System.out.println("Request ID: " + e.getRequestId());
        }


        //===========================================================//
        //================= Canceling the Request =================//
235     //===========================================================//

        try {
          // Cancel requests.
          CancelSpotInstanceRequestsRequest cancelRequest = new CancelSpotIn
stanceRequestsRequest(spotInstanceRequestIds);
240       ec2.cancelSpotInstanceRequests(cancelRequest);
        } catch (AmazonServiceException e) {
          // Write out any exceptions that may have occurred.
          System.out.println("Error canceling instances");
          System.out.println("Caught Exception: " + e.getMessage());
```

```
245        System.out.println("Reponse Status Code: " + e.getStatusCode());
           System.out.println("Error Code: " + e.getErrorCode());
           System.out.println("Request ID: " + e.getRequestId());
        }

250     //=========================================================//
        //============== Terminating any Instances ================//
        //=========================================================//
        try {
           // Terminate instances.
255        TerminateInstancesRequest terminateRequest = new TerminateInstances
Request(instanceIds);
           ec2.terminateInstances(terminateRequest);
        } catch (AmazonServiceException e) {
           // Write out any exceptions that may have occurred.
           System.out.println("Error terminating instances");
260        System.out.println("Caught Exception: " + e.getMessage());
           System.out.println("Reponse Status Code: " + e.getStatusCode());
           System.out.println("Error Code: " + e.getErrorCode());
           System.out.println("Request ID: " + e.getRequestId());
        }
265   } // main
    }
```

Tags are a simple first step toward making it easier to manage your own cluster of instances. To read more about tagging Amazon EC2 resources, go to Using Tags in the Amazon Elastic Compute Cloud User Guide.

Bringing It All Together

To bring this all together, we provide a more object-oriented approach that combines the steps we showed in this tutorial into one easy to use class. We instantiate a class called `Requests` that performs these actions. We also create a `GettingStartedApp` class, which has a main method where we perform the high level function calls.

The complete source code is available for download at GitHub.

Congratulations! You've completed the Advanced Request Features tutorial for developing Spot Instance software with the AWS SDK for Java.

Reserved Instances

You can use Amazon Elastic Compute Cloud (Amazon EC2) Reserved Instances to take advantage of elastic cloud computing while lowering costs and reserving capacity. With Reserved Instances, you pay a low, one-time fee to reserve capacity for a specific instance and get a significant discount on the hourly fee for that instance when you use it. Reserved Instances can provide substantial savings over owning your own hardware or running only On-Demand instances. At the same time, you know that the capacity you need is available to you when you require it.

> **Note**
>
> If you are using the Amazon Web Services (AWS) free usage tier to run Amazon EC2 instances, and then you purchase a Reserved Instance, you will be charged for the Reserved Instance under standard pricing guidelines. The AWS free usage tier is available for new AWS accounts. The free tier applies to certain participating AWS services up to a specific maximum amount of usage each month, for a period of a year. When an account goes over the usage limit for a free tier, the standard AWS service rates will be billed to your credit card. For information about the free tier and the applicable services and usage amounts, go to AWS Free Usage Tier.

Reserved Instances are available in three varieties—Heavy Utilization, Medium Utilization, and Light Utilization—that enable you to optimize your Amazon EC2 costs based on your expected utilization. You can start lowering your costs today by matching your application's usage pattern with the pricing model that works best for you.

To purchase an Amazon EC2 Reserved Instance, you must select an instance type (such as m1.small), platform (Linux/UNIX, Windows, Windows with SQL Server, or SUSE), location (Region and Availability Zone), and term (either one year or three years). If you want to use Amazon Virtual Private Cloud (Amazon VPC), you must specify that you want your capacity reservation allocated to Amazon VPC.

If you're using Amazon VPC, you must select a platform that includes *Amazon VPC* in its name. You can also choose to purchase Reserved Instances that are physically isolated at the host hardware level by specifying *Dedicated* as the instance tenancy. For more information about Dedicated Instances, go to Using EC2 Dedicated Instances Within Your VPC in the *Amazon Virtual Private Cloud User Guide*.

These sections describe Reserved Instance basics:

To understand consolidated billing and how the pricing benefit of Reserved Instances is applied, see Understanding the Pricing Benefit and Consolidated Billing (p. 200).

To understand Reserved Instance Pricing Tiers and how to take advantage of the discount pricing, see Understanding Reserved Instance Pricing Tiers (p. 203).

If you want to get started with Reserved Instances, see Working with Reserved Instances (p. 211).

Reserved Instances Offerings

You can choose a Reserved Instance type based on how you plan to utilize your instance. You can pick the fee structure that fits your needs.

Starting with API version 2011-11-01, you can choose from a variety of Reserved Instance offerings that address your projected utilization of the instance: *Heavy Utilization*, *Medium Utilization*, and *Light Utilization*.

Heavy Utilization Reserved Instances enable workloads that have a consistent baseline of capacity or run steady-state workloads. Heavy Utilization Reserved Instances require the highest upfront commitment, but if you plan to run more than 79 percent of the Reserved Instance term you can earn the largest savings (up to 58 percent off of the On-Demand price). Unlike the other Reserved Instances, with Heavy Utilization Reserved Instances, you pay a one-time fee, followed by a lower hourly fee for the duration of the term *regardless* of whether or not your instance is running.

Medium Utilization Reserved Instances are the best option if you plan to leverage your Reserved Instances a substantial amount of the time, but want either a lower one-time fee or the flexibility to stop paying for your instance when you shut it off. This offering type is equivalent to the Reserved Instance offering available before API version 2011-11-01. Medium Utilization Reserved Instances are a more cost-effective option when you plan to run more than 40 percent of the Reserved Instance term. This option can save you up to 49 percent off of the On-Demand price. With Medium Utilization Reserved Instances, you pay a slightly higher one-time fee than with Light Utilization Reserved Instances, and you receive lower hourly usage rates when you run an instance.

Light Utilization Reserved Instances are ideal for periodic workloads that run only a couple of hours a day or a few days per week. Some of these use cases, such as disaster recovery, also require reserved capacity to meet potential demand without notice. Using Light Utilization Reserved Instances, you pay a one-time fee followed by a discounted hourly usage fee when your instance is running. You can start saving when your instance is running more than 17 percent of the Reserved Instance term, and you can save up to 33 percent off of the On-Demand rates over the entire term of your Reserved Instance.

Remember that discounted usage fees for Reserved Instance purchases are tied to instance type and Availability Zone. If you shut down a running EC2 instance on which you have been getting a discounted rate as a result of a Reserved Instance purchase, and the term of the Reserved Instance has not yet expired, you will continue to get the discounted rate if you launch another instance with the same specifications during the term.

The following table summarizes the differences between the Reserved Instances offering types.

Reserved Instance Offerings

Offering	Upfront Cost	Usage Fee	Advantage
Heavy Utilization	Highest	Lowest hourly fee. Applied to the whole term whether or not you're using the Reserved Instance.	Lowest overall cost if you plan to utilize your Reserved Instances more than 76 percent of a 3-year term.
Medium Utilization	Average	Hourly usage fee charged for each hour you use the instance. We encourage you to turn off your instances when you aren't using them so you won't be charged for them.	Suitable for elastic workloads or when you expect moderate usage, more than 44 percent of a 3-year term.

Offering	Upfront Cost	Usage Fee	Advantage
Light Utilization	Lowest	Hourly usage fee charged. Highest fees of all the offering types, but they apply only when you're using the Reserved Instance. We strongly encourage you to turn off your instances when you aren't using them so you won't be charged for them.	Highest overall cost if you plan to run all of the time, however lowest overall cost if you anticipate you will use your Reserved Instances infrequently, more than about 8 percent of a 3-year term.

Checklist for Finding, Purchasing, Verifying, and Running Reserved Instances

The following list describes the process for using Reserved Instances. You can use the AWS Management Console (p. 211), the Command Line Tools (p. 213), or the API (p. 214) to perform any of these tasks.

1. Specify the details about the Reserved Instance you want to launch.
   - The platform on which to run your instance.
   - The instance type of the instance.
   - The term (time period) over which you want to hold the instance.
   - The tenancy specification, if you want to reserve capacity for your instance to run in single-tenant hardware.
   - The region and Availability Zone where you want to run the instance.

2. Choose the offering type that addresses your cost and payment needs.
   - Heavy Utilization
   - Medium Utilization
   - Light Utilization

   For more information about these offering types, see Reserved Instances Offerings (p. 195).

3. Search for offerings that meet the criteria you specified.
   If you want to use a Reserved Instance with Amazon Virtual Private Cloud (Amazon VPC), make sure to search for and select an offering that includes *Amazon VPC* in the description.

4. Purchase offerings that fulfill your requirements.

5. Run your Reserved Instance by launching an instance with the criteria you specified when you purchased your Reserved Instance. These criteria include platform, instance type, region, and Availability Zone.

6. Confirm that your instances are running as specified.

   Note

   If your Reserved Instance purchase crosses from one pricing tier to another, your new purchase will go into a pending state while the Reserved Instance service processes the purchase. Processing involves purchasing two or more different Reserved Instances for you: An amount at one tier, and an amount at the next tier which would be either the discounted rate or the higher discounted rate. As a result, the Reserved Instance ID returned by your purchase CLI command and API action will be different from the actual ID of the new Reserved Instances. For more information, see Understanding Reserved Instance Pricing Tiers (p. 203).

If you are new to Reserved Instances, and you want to learn about the tools need to work with Reserved Instances, see Tools and Prerequisites (p. 198).

To get started finding and purchasing your Reserved Instance, see Working with Reserved Instances (p. 211).

Tools and Prerequisites

To work with Reserved Instances, we assume you have read and completed the instructions described in Getting Started with EC2, which provides information on creating your Amazon EC2 account and credentials.

You can use the AWS Management Console (p. 211), the Command Line Tools (p. 213), or the API (p. 214) to work with Amazon EC2 Reserved Instances and search for offerings.

If you use the Amazon EC2 command line interface (CLI) tools, we assume that you have read and completed the instructions described in Getting Started with the Command Line Tools. This topic walks you through setting up your environment for use with the CLI tools.

In addition, whichever you choose to use—AWS Management Console, the CLI, or API—Amazon EC2 provides tools that help you to identify Reserved Instances offerings, purchase your Reserved Instances, and manage them.

AWS Management Console

The AWS Management Console has tools specifically designed for Reserved Instances tasks. The console also has general tools that you can use to manage the instances launched when you use your Reserved Instances. The following list identifies the tools you can use in the AWS Management Console:

- The **Reserved Instances** page is where you interact with your Reserved Instances.
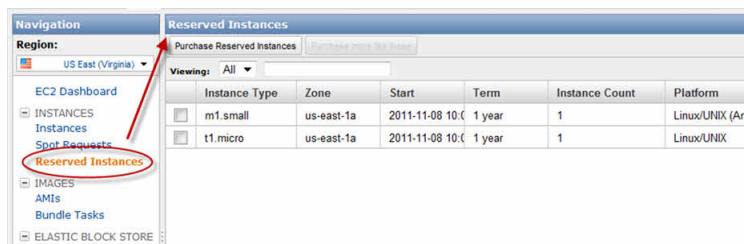
- Use the **Purchase Reserved Instances** page to specify the details of Reserved Instances you want to purchase.
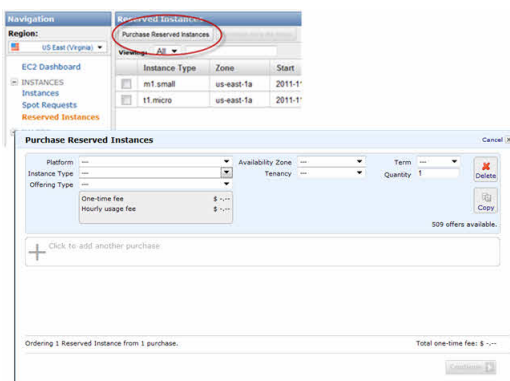
- Use the **Instances** page to manage the instances you launched using your Reserved Instances.
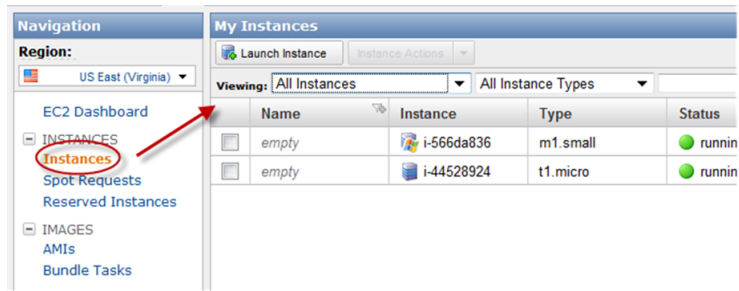
Command Line Tools

To purchase Reserved Instances, you use Amazon EC2 command line interface (CLI) tools specifically designed for these tasks. To manage the instances when your Reserved Instances are launched, use the same CLI commands that you use for Amazon EC2 instances.

The following table lists the CLI commands you use for Reserved Instances tasks.

Task	CLI
List Reserved Instances that you have purchased.	`ec2-describe-reserved-instances`
View the Reserved Instances offerings that are available for purchase.	`ec2-describe-reserved-instances-offerings`
Purchase a Reserved Instance.	`ec2-purchase-reserved-instances-offering`

For information about CLI commands, go to the Amazon Elastic Compute Cloud Command Line Reference.

API

To purchase Reserved Instances, you use API calls specifically designed for these tasks. To manage the instances when your Reserved Instances are launched, use the same API calls that you use for Amazon EC2 instances.

The following table lists the API calls you use for Reserved Instances tasks.

Task	API
List Reserved Instances that you have purchased.	`DescribeReservedInstances`
View the Reserved Instances offerings that are available for purchase.	`DescribeReservedInstancesOfferings`
Purchase a Reserved Instance.	`PurchaseReservedInstancesOffering`

For information about API actions, go to the Amazon Elastic Compute Cloud API Reference.

Understanding the Pricing Benefit and Consolidated Billing

Topics

When you purchase Reserved Instances, you get two benefits: a capacity reservation for a number of instances you will launch at some future time and a discounted hourly usage fee. The rest of the experience of launching and working with instances remains the same as with On-Demand instances.

This section discusses how Amazon EC2 applies the pricing benefit of Reserved Instances. You'll also see how consolidated billing works.

Applying the Pricing Benefit of Reserved Instances

With Reserved Instances, you pay an upfront fee for the capacity reservation to launch Amazon EC2 instances to the specifications you defined during the purchase. After you purchase Reserved Instances, you cannot change the specifications (product platform, instance type, Availability Zone, etc.).

In addition to the capacity reservation, you also get a discounted rate on the hourly fee for running On-Demand EC2 instances associated with the same account that purchased the Reserved Instances. For the discount to apply, the On-Demand instances must match the specifications for the Reserved Instances.

For example, let's say User A is running the following On-Demand EC2 instances:

- (4) m1.small instances in Availability Zone us-east-1a
- (4) c1.medium instances in Availability Zone us-east-1b
- (2) c1.xlarge instances in Availability Zone us-east-1b

Then User A purchases the following Medium Utilization Reserved Instances:

- (2) m1.small instances in Availability Zone us-east-1a
- (3) c1.medium instances in Availability Zone us-east-1a
- (1) c1.xlarge instance in Availability Zone us-east-1b

In purchasing the Reserved Instances, User A pays an upfront fee for the capacity reservation so he can launch the six instances to his specifications when he needs them. In addition, he is also getting a discount on the hourly usage fees for the equivalent of six instances each month. Since he already has instances running when he purchases the Reserved Instances, Amazon EC2 will automatically apply the discounted hourly rates to the already running On-Demand instances that match the specfications for the Reserved Instances he purchased.

This is what happens:

- Amazon EC2 applies the discounted usage fee rate for two m1.small Reserved Instances that he purchased to two of the four running Amazon EC2 instances in Availability Zone us-east-1a.
  The other two EC2 instances in Availability Zone us-east-1a will be charged at the current On-Demand rate.

- Amazon EC2 doesn't apply discounted rates from the three c1.medium Reserved Instances because the c1.medium Reserved Instances are in a different Availability Zone from the running c1.medium Amazon EC2 instances.

  The four running c1.medium Amazon EC2 instances will be charged at the current On-Demand rate.

  If User A launches a c1.medium EC2 instance in Availability Zone us-east-1a, Amazon EC2 will apply the Reserved Instance discounted usage fee rate to that instance.

- Amazon EC2 applies the discounted usage fee rate for one c1.xlarge Reserved Instance that he purchased to one of the two running Amazon EC2 instances in Availability Zone us-east-1b.

  The other c1.xlarge EC2 instance in Availability Zone us-east-1b will be charged at the current On-Demand rate.

In this example scenario, by purchasing the six Reserved Instances, User A saves on the hourly fee charged against two m1.small and one c1.xlarge On-Demand EC2 instances he had already running. At the same time, he is assured of the capacity to run the six instances when he needs them.

Reserved Instances and Consolidated Billing

The pricing benefits of Reserved Instances are shared when the purchasing account is part of a set of accounts billed under one Consolidated Billing Payer account. Consolidated billing allows you to pay all of your charges via one Payer account (the one that will pay the bill). The amount of hourly usage for each month across all child accounts is also aggregated in the parent account. This billing is typically useful for companies in which there are different functional teams or groups.

For Reserved Instances, the amount of usage across all linked accounts is also aggregated in the payer account, by the hour for each month, and then the normal Reserved Instance logic is applied to calculate the bill. The discount is spread across the consolidated account. The allocation of the total cost is determined by the ratio of each child account's usage divided by the total usage of the parent account. However, it is important to note that the capacity reservation remains with the purchasing account. Keep in mind that capacity reservation only applies to the product platform, instance type, and Availability Zone specified in the purchase.

If the account is part of a consolidated billing account, the upfront cost of the Reserved Instance purchases is paid by the Payer account. Additionally, all of the subsequent usage charges will be paid by the payer account as long as the account remains linked with the payer account. However, all of the pricing benefit will also be shared with all of the linked accounts as long as it remains linked to the payer account.

Consolidated billing accounts also benefit from Reserved Instance Pricing Tiers. For more information, see .

To illustrate, we'll expand on the previous example. For the purpose of explaining how billing works, we're simplifying the scenario with the following assumptions:

- All instances are of the same instance type, running on the same product platform in the same Availability Zone.
- The usage fee cost per hour for each instance is $1.00.
- The discounted usage fee cost per hour for each instance is $0.50.
- User A, User B, and User C, fall under account CB AZ.
- In one month, the total number of instances running across all three users is 18 instances, all at the same rate.
- There are 30 days in a month, and the instances run 10 hours a day. This is a total of 300 hours a month.
- User A, the purchaser of the six Reserved Instances, is running 9 Instances, which is one-half of the total instances billed to the master account.

- User B is running 6 Instances, which is one-third of the total instances billed.

- User C is running 3 Instances, which is one-sixth of the total instances billed.

The following table shows how the consolidated bill would look without the benefit of Reserved Instances purchased by User A.

> **Note**
>
> Amazon EC2 doesn't determine or allocate Reserved Instances usage to a particular account. Instead, Amazon EC2 adds up the usage, calculates cost, and divides the total charges across each account.

Consolidated Billing Without Reserved Instances Purchased

User	Instance Used	Calculated Usage Fee Cost
User A	9 Instances Comprises one-half of total 18 instances	9 (instances) X 300 (hours for the month) = $2,700.00
User B	6 Instances Comprises one-third of total 18 instances	6 (instances) X 300 (hours for the month) = $1,800.00
User C	3 Instances Comprises one-sixth of total 18 instances	3 (instances) X 300 (hours for the month) = $900.00

With User A's purchase of Reserved Instances, the benefit of the discounted rate for six Reserved Instances will be divided across the three accounts under the master account CB AZ. The division will be based on the usage rate:

- User A: one-half of 6 = 3 instances

- User B: one-third of 6 = 2 instances

- User C: one-sixth of 6 = 1 instance

Master account CB AZ will save a total of $900 in usage fees after the Reserved Instances discounts are applied. The following table shows how the consolidated bill would look with the benefit of Reserved Instances purchased by User A.

Consolidated Billing Showing Benefits of Reserved Instances Purchased

User	Instance Used (with Reserved Instance Usage Fee Discount)	Calculated Usage Fee Cost	Savings
User A	9 Instances Comprises one-half of total 18 instances Gets one-half of discount for 6 Reserved Instances applied	3 (instances) X 300 (hours for the month) X $0.50 (Reserved Instance rate) = $450.00 6 (instances) X 300 (hours for the month) X $1.00= $1,800.00 Total cost: $2,250.00	$450.00
User B	6 Instances Comprises one-third of total 18 instances Gets one-third of discount for 6 Reserved Instances applied	2 (instances) X 300 (hours for the month) X $0.50 (Reserved Instance rate) = $300.00 4 (instances) X 300 (hours for the month) = $1,200.00 Total cost: $1,500.00	$300.00
User C	3 Instances Comprises one-sixth of total 18 instances Gets one-sixth of discount for 6 Reserved Instances applied	1 (instance) X 300 (hours for the month) X $0.50 (Reserved Instance rate) = $150.00 2 (instances) X 300 (hours for the month) = $600.00 Total cost: $750.00	$150.00

Note

There are different pricing options for Reserved Instances, depending on the instance type, Availability Zone, term period, and product platform. For information about pricing, see the Amazon EC2 Reserved Instances product page.

Understanding Reserved Instance Pricing Tiers

You can use Amazon EC2 Reserved Instance Pricing Tiers to reduce your computing costs as your business grows. With Reserved Instance Pricing Tiers, you are entitled to a discount on your upfront fees and your usage fees when the total list price of your Reserved Instance upfront fees per Region goes over the discount threshold. *List price* is the amount AWS charges at the undiscounted tier level at the time of purchase. *List price* is different from *paid price*. *Paid price* is the actual amount that an account pays for the Reserved Instances. If the account qualifies for a discount, the *paid price* will be lower than the *list price*. *Paid price* is the same as *fixed price* when you work with Reserved Instance tools in the AWS Management Console, command line interface (CLI), and the API.

This section discusses key concepts you need to understand to take advantage of the discount pricing that is built into the Reserved Instance Pricing Tiers.

-

Reserved Instance Pricing Tiers

The following table shows the total list price of the upfront fees for active Reserved Instances per Region and the discount that is applied at each tier.

Tier level	Total list price of upfront fees for active Reserved Instances per Region	Discount applied to upfront and usage fees
Tier 0	$0 - $250,000	Standard Reserved Instance upfront and usage fees
Tier 1	$250,000 - $2,000,000	10 percent discount
Tier 2	$2,000,000 - $5,000,000	20 percent discount
Tier 3	Over $5,000,000	Contact Us

How Reserved Instance Pricing Tiers Work

You can receive discounts on your purchases of future Reserved Instances based on the total list price of the upfront fees for active Reserved Instances you have for each Amazon Web Services (AWS) Region. When the total list price goes above certain price points, any future Reserved Instances purchased will be charged at a discounted rate (as long as your total list price stays above that price point). If a single purchase takes you over the threshold, then the portion of that purchase that is above the price threshold will be charged at the discounted rate.

Here's an example: Let's assume you currently have $200,000 worth of active Reserved Instances in the us-east-1 Region. You purchase 75 Reserved Instances at the list price of $1000 each. That's a total of $75,000, which brings the total amount you have paid for active Reserved Instances to $275,000. Since the discount pricing threshold is $250,000, the first $50,000 of your new purchase would not receive a discount. The remaining $25,000, which exceeds the discount pricing threshold, would be discounted by 10 percent ($2,500). This means you will only be charged $22,500 for the remainder of your purchase (25 instances), and you will be charged discounted usage rates for those 25 Reserved Instances.

If your account is part of a consolidated billing account, you can benefit from the Reserved Instance Pricing Tiers. A consolidated billing account aggregates into a single list price the list prices of all of the active Reserved Instances accounts in a Region that are part of the consolidated billing account. When the total list price of active Reserved Instances for the consolidated billing accounts reaches the discounted tier level, any Reserved Instances purchased after this point by a member of the consolidated account will be charged at the discounted rate (for as long as the total list price for that consolidated account stays above that price point).

Here's how this works: Let's assume that two accounts—A and B—are part of a consolidated billing account. All the active Reserved Instances in the consolidated billing account are in one region. Account A has Reserved Instances worth $135,000; Account B has Reserved Instances worth $115,000. The total upfront cost of the consolidated bill of Accounts A and B is $250,000. Remember, $250,000 is the discount pricing threshold. This means that when either or both of the A and B accounts purchase additional Reserved Instances, the cost of the new purchases will be discounted by 10 percent. So, when account B purchases Reserved Instances at a list price of $15,000, the consolidated account will only be charged $13,500 for the new Reserved Instances ($15,000 minus the 10 percent discount of $1500 equals $13,500), and account B will be charged discounted usage rates for those new Reserved Instances.

For more information about how the benefits of Reserved Instances apply to consolidated billing accounts, see Reserved Instances and Consolidated Billing (p. 201).

Determining Your Pricing Tier Level

You can learn which Reserved Instance pricing tier your account is in by determining the total *list price* of the purchases of your account per Region. *List price* is the amount charged at the undiscounted tier at the time of purchase. *List price* is different from *paid price*. *Paid price* is the actual amount that an account paid for the Reserved Instances. The *paid price* will be either the *list price* or the discounted price. This means that if you have Reserved Instances that qualified for purchase at the 10 percent discount tier, the *paid price* will be 10 percent lower than the *list price*.

The *fixed price* value—displayed in the AWS Management Console, the CLI, and the API—shows the price you paid (*paid price*) per instance for all active Reserved Instances you purchased. To determine the Reserved Instance pricing tier for your account, get the total for all the upfront fees (*fixed price*) in a Region. Typically, you can use the total value as basis for determining your tier level.

If you have already crossed a tier, you may qualify for a higher tier than the previous process may suggest. The most accurate approach available to determine the Reserved Instance Pricing Tiers that you qualify for requires you to look at the current *list price* for each Reserved Instance by calling the `ec2-describe-reserved-instances-offerings` command in the CLI or the `DescribeReservedInstancesOfferings` action in the API. Take the maximum value of the list price from the output of the command or action compared to the fixed price (i.e., *paid price*) that is returned by the `ec2-describe-reserved-instances` CLI command or `DescribeReservedInstances` API action.

The reason this comparison yields better information is that Amazon Web Services lowered prices and introduced pricing tiers in March 2012. Purchase prices before the introduction of pricing tiers were higher than current prices. So any Reserved Instances purchased before the introduction of pricing tiers will have the same *list price* and *paid price*. Therefore, when you take the maximum of the current *list price* and *fixed price*, you will get the actual list price for the offerings.

Alternatively, you can determine your tier level by looking at your bills and using the spreadsheet to add up all the list prices.

The following list shows how the *fixed price* values are returned when you use the AWS Management Console, the CLI, and the API:

- AWS Management Console

  In the AWS Management Console, use the **Reserved Instances** page of the Amazon EC2 console to find out the pricing tier in which your account belongs. The **Fixed Price** column shows the amount you paid for each of your active Reserved Instances.

- Command Line Interface (CLI)

  Use the `ec2-describe-reserved-instances` command to find out the pricing tier in which your account belongs. The `ec2-describe-reserved-instances` command will return `FixedPrice` information showing the amount you paid for all your active Reserved Instances.

  Your `ec2-describe-reserved-instances` command should look like the following example:

```
PROMPT> ec2-describe-reserved-instances --headers
```

Amazon EC2 returns output similar to the following example:

```
PROMPT> ec2-describe-reserved-instances
Type ReservedInstancesId AvailabilityZone InstanceType ProductDescription
Duration FixedPrice UsagePrice InstanceCount Start State Currency InstanceTen
ancy OfferingType
RESERVEDINSTANCES f127bd27-f0e9-43bb-89f5-1b8c030bc8f9 us-east-1b m1.small
Linux/UNIX 1y  227.5 0.03 1 2011-11-28T16:17:12+0000 default active USD Medium
 Utilization
RESERVEDINSTANCES f127bd27-f62e-4763-9e09-ee24f8ccef5d us-east-1b m1.large
Windows with SQL Server 3y  3400.0 0.35 1 2011-12-02T06:13:25+0000 default
active USD Medium Utilization
RESERVEDINSTANCES 1ba8e2e3-e8b2-4637-a124-ec9c11495ac9 us-east-1d m1.small
Linux/UNIX (Amazon VPC) 1y  280.0 0.035 1 2011-12-02T06:05:28+0000 dedicated
 active USD Medium Utilization
RESERVEDINSTANCES 46a408c7-89fe-4b02-bb5e-ecb9bbc510eb us-east-1d m2.xlarge
Linux/UNIX 1y  1000.0 0.5 1 2011-12-02T06:05:27+0000 default active USD Lower
 Utilization
RESERVEDINSTANCES af9f760e-96ae-4d12-8abe-8eb1e7a2bbdb us-east-1d t1.micro
Linux/UNIX 1y  54.0 0.0070 10 2011-12-02T06:12:09+0000 default active USD Me
dium Utilization
RESERVEDINSTANCES 46a408c7-ae07-4611-9d1c-f6d3a947f8d3 us-east-1a m1.small
Linux/UNIX (Amazon VPC) 1y  227.5 0.03 1 2011-11-08T18:00:02+0000 default
active USD Medium Utilization
RESERVEDINSTANCES bbcd9749-1211-4134-a7e7-0cdfec1caca5 us-east-1a t1.micro
Linux/UNIX 1y  54.0 0.0070 1 2011-11-08T18:03:20+0000 default active USD Medium
 Utilization
RESERVEDINSTANCES d16f7a91-556f-4db5-afc9-4dd0673334c6 us-east-1a m1.large
Windows with SQL Server 3y  3400.0 0.35 1 2011-12-02T06:22:03+0000 default
active USD Medium Utilization
REQUEST ID d9121e8b-e7c1-49f2-88cd-b478cce99751
```

- API
  Use the `DescribeReservedInstances` action to find out the pricing tier in which your account belongs.
  The `DescribeReservedInstances` action will return `fixedPrice` information showing the amount
  you paid for all your active Reserved Instances.

  Your `DescribeReservedInstances` action should look like the following example:

```
https://ec2.amazonaws.com/
?Action=DescribeReservedInstances
&...auth parameters...
```

Amazon EC2 returns output similar to the following example:

```
<DescribeReservedInstances xmlns="http://ec2.amazonaws.com/doc/2012-07-20/">

    <reservedInstancesSet>
      <item>
       <reservedInstancesId>f127bd27-f0e9-43bb-89f5-1b8c030bc8f9</reserved
InstancesId>
          <instanceType>m1.small</instanceType>
          <availabilityZone>us-east-1b</availabilityZone>
```

```
              <start>2011-11-28T16:17:12.845Z</start>
              <duration>31536000</duration>
              <fixedPrice>227.5</fixedPrice>
              <usagePrice>0.03</usagePrice>
              <instanceCount>1</instanceCount>
              <productDescription>Linux/UNIX</productDescription>
              <state>active</state>
              <instanceTenancy>default</instanceTenancy>
              <currencyCode>USD</currencyCode>
              <offeringType>Medium Utilization</offeringType>
              <recurringCharges/>
          </item>
          ...
      </reservedInstancesSet>
      </DescribeReservedInstances>
```

Purchasing at a Discount Tier Price

The Amazon EC2 Reserved Instance service will automatically apply any discounts to that part of your Reserved Instance purchase that falls within a discounted tier.

Using the AWS Management Console, CLI, or API, you proceed through your purchase in the same way you do now.

- AWS Management Console: Use the **Purchase Reserved Instances** button on the **Reserved Instances** page of the Amazon EC2 console.
- Command Line Interface (CLI): Use the `ec2-purchase-reserved-instances-offering` command.
- API: Use the `PurchaseReservedInstancesOffering` call.

If your purchase crosses into a discounted pricing tier, the AWS Management Console, the `ec2-describe-reserved-instances` command, or the `DescribeReservedInstances` action will show multiple entries for that purchase—an entry for that part of the purchase that will be charged the regular Reserved Instance price, and another entry or entries for that part of the purchase that will be charged the discounted rate that applies.

For example, let's say that you have a list price total of $245,000 in Reserved Instances in a Region. You want to purchase more Reserved Instances at a total list price of $10,000. You purchase these Reserved Instances in the same way you've done before. Remember that the threshold for the 10 percent discount tier is $250,000, so part of your purchase crosses into the discount tier. When the transaction is complete, the list you'll get from the AWS Management Console, the CLI, or the API will show two reservations—a reservation for $5,000 at the undiscounted rate, and another for $4,500, which is the discounted price. The discount is 10 percent of $5,000 or $500.

The pricing discount applies to future purchases after the total list price cost of your active Reserved Instances reaches the discounted pricing tier. However, if some of your Reserved Instances expire and the total list price for your active Reserved Instances fall below the discounted pricing tier level, then the next purchase you make will be at the retail rates for Reserved Instances. To illustrate, let's use the previous example: You currently have a list price total of $254,500 in Reserved Instances. $250,000 is at the undiscounted rate, and $4,500 is at the discounted rate. In two months, some of your Reserved Instances expire, and your total is back to $234,500, which is below the threshold for the 10 percent discount tier. When you purchase $15,000 in Reserved Instances, you will be charged the retail rate. However, the $4,500 in Reserved Instances that you purchased earlier at the discounted rate will continue to be charged at the discounted rate.

The main point to keep in mind is that *list price* is the undiscounted price of the Reserved Instance at the time of purchase, while the *fixed price* value is the *paid price* of all Reserved Instances purchased. The discount tier is based on *list price*.

In addition, when you purchase Reserved Instances, one of three possible scenarios will occur:

* Your purchase of Reserved Instances per Region is still within the lowest tier (tier 0). This means that your purchase does not reach the threshold for the discount tiers, and you don't get a discount. The process works as it does today.
* Your purchase of Reserved Instances within a Region crosses the threshold of the first discount tier. This means that the newly purchased Reserved Instances will go into the pending state. For this purchase, the Reserved Instances service will purchase Reserved Instances for you at two different rates: An amount at the undiscounted rate, and an amount at the discounted rate. It is important to understand that the Reserved Instance IDs you get back from the CLI or API will be different from the new Reserved Instance IDs that will actually be created.
* Your entire purchase of Reserved Instances within a Region is completely within a discount tier. This means that the newly purchased Reserved Instances will go into the pending state. For this purchase, the Reserved Instances service will purchase Reserved Instances at the appropriate discount level for you. It is important to understand that, as with the previous scenario, the Reserved Instance IDs you get back from the CLI or API will be different from the new Reserved Instance IDs that will actually be created.

Reserved Instance IDs

When your total purchase crosses one or more Reserved Instance discount pricing tiers, the Reserved Instance IDs returned by your purchase command can be different from the Reserved Instance IDs returned by the describe command that you call after the purchase is complete.

Using the CLI, here's an example output of the `ec2-describe-reserved-instances` command if you use the Reserved Instance ID `1ba8e2e3-edf1-43c3-b587-7742bc77b9ba`, which was returned by `ec2-purchase-reserved-instances-offering`.

```
$ ec2-describe-reserved-instances -H --region sa-east-1 1ba8e2e3-edf1-43c3-b587-
7742bc77b9ba
Type ReservedInstancesId AvailabilityZone InstanceType ProductDescription Dura
tion FixedPrice UsagePrice InstanceCount Start State Currency InstanceTenancy
OfferingType
```

The describe command will not recognize the Reserved Instance ID.

What really happened is that the Reserved Instance service generated several Reserved Instance IDs because your purchase crossed from the undiscounted tier (tier 0) to the first discounted tier (tier 1). The portion of Reserved Instances that is in tier 1 has a 10 percent discount applied to the *list price*.

Your entire purchase would look like the following example:

```
$ ec2-describe-reserved-instances -H --region sa-east-1 bbcd9749-05f0-4ada-96c8-
812f5f0ab9b3
Type ReservedInstancesId AvailabilityZone InstanceType ProductDescription Dura
tion FixedPrice UsagePrice InstanceCount Start State Currency InstanceTenancy
OfferingType
RESERVEDINSTANCES bbcd9749-05f0-4ada-96c8-812f5f0ab9b3 sa-east-1a t1.micro
Linux/UNIX 3y  20000.0 0.0090 2 2012-03-02T23:20:16+0000 default payment-pending
 USD Medium Utilization
$ ec2-describe-reserved-instances -H --region sa-east-1 1ba8e2e3-346e-4e5b-a2e2-
b559243f2325
```

```
Type ReservedInstancesId AvailabilityZone InstanceType ProductDescription Dura
tion FixedPrice UsagePrice InstanceCount Start State Currency InstanceTenancy
OfferingType
RESERVEDINSTANCES 1ba8e2e3-346e-4e5b-a2e2-b559243f2325 sa-east-1a t1.micro
Linux/UNIX 3y  18000.0 0.0080 3 2012-03-02T23:20:17+0000 default payment-pending
 USD Medium Utilization
$ ec2-describe-reserved-instances -H --region sa-east-1 af9f760e-868c-48f4-87e2-
44576dbf05ef
tType ReservedInstancesId AvailabilityZone InstanceType ProductDescription
Duration FixedPrice UsagePrice InstanceCount Start State Currency InstanceTenancy
 OfferingType
RESERVEDINSTANCES af9f760e-868c-48f4-87e2-44576dbf05ef sa-east-1a t1.micro
Linux/UNIX 3y  18000.0 0.0080 5 2012-03-02T23:20:18+0000 default payment-pending
 USD Medium Utilization
```

Reserved Instance ID `bbcd9749-05f0-4ada-96c8-812f5f0ab9b3` is the ID for the Reserved Instances
that you purchased at the undiscounted *list price* of $20,000 each. Reserved Instance IDs
`1ba8e2e3-346e-4e5b-a2e2-b559243f2325` and `af9f760e-868c-48f4-87e2-44576dbf05ef` are
the IDs for the Reserved Instances that you purchased at the 10 percent discount rate ($20,000 minus
$2,000).

Scenario Showing Purchases that Cross Pricing Tiers

Let's walk through an example scenario in which your purchases cross the various pricing tiers.

Two months ago, you purchased 100 Reserved Instances in the us-east-1a Region at $2000 each. That
purchase totaled $200,000. The *list price* for this purchase is $2000. The amount you paid was $2000
per Reserved Instance, so it is the *paid price*, and the paid price is the value that will be shown under
`fixed-price`. Your purchases are still within the first, undiscounted tier.

The following table illustrates this example.

Purchase Number	List Price	Amount Paid	Fixed Price	Total RI Purchased	Total List Price Cost	Total Paid Amount
Purchase 1	$2000	$2000	$2000	100	$200,000	$200,000

Later you want to purchase more Reserved Instances. Let's say that AWS lowered prices and the same
type of Reserved Instance now is available at $1000 each. You purchase 75 of these Reserved Instances.
The *List Price* for this purchase is $1000. The purchase of 75 Reserved Instances at $1000 each totals
$75,000. This raises the total cost of your active Reserved Instances to $275,000. The threshold for the
discount tier is $250,000. This purchase crosses you into the discount tier, also called tier 1 in the Reserved
Instance Pricing Tiers (p. 204) table.

In this discount tier, you get a 10 percent discount on all your purchases of Reserved Instances in the
same Region above $250,000. So, you will pay the new list price of $1000 each for the first 50 Reserved
Instances (total amount paid of $50,000). And you will pay $900 each—the $1000 list price minus the 10
percent discount— for the remaining 25 Reserved Instances (total amount paid of $22,500). Your *fixed
price* and the *price paid* for the discounted Reserved Instances will both show $900.

The following table illustrates this example.

Purchase Number	List Price	Amount Paid	Fixed Price	Total RI Purchased	Total List Price Cost	Total Paid Amount
Purchase 1	$2000	$2000	$2000	100	$200,000	$200,000

Purchase Number	List Price	Amount Paid	Fixed Price	Total RI Purchased	Total List Price Cost	Total Paid Amount
Purchase 2	$1000	$1000	$1000	50	$50,000	$50,000
	$1000	$900	$900	25	$25,000	$22,500
Totals					$275,000	$272,500

Six months later, let's assume that your business experiences tremendous growth, and you need to purchase 1800 additional Reserved Instances in the same Region. At a list price of $1000, this purchase will total $1,800,000. When you add this new purchase to your previous purchases of already active Reserved Instances that total $272,500, your new total will be $2,072,500. This new total crosses the threshold for the next discount tier (tier 2 in the Reserved Instance Pricing Tiers (p. 204) table). In this tier, discounts of 20 percent apply to purchases above $2,000,000.

Your new purchase will be charged two different discount rates: The 1725 Reserved Instances that fall within tier 1 will be discounted 10 percent. The remaining 75 Reserved Instances that put the total list price cost above $2,000,000, and thus in tier 2, will be discounted 20 percent.

Purchase Number	List Price	Amount Paid	Fixed Price	Total RI Purchased	Total List Price Cost	Total Paid Amount
Purchase 1	$2000	$2000	$2000	100	$200,000	$200,000
Purchase 2	$1000	$1000	$1000	50	$50,000	$50,000
	$1000	$900	$900	25	$25,000	$22,500
Purchase 3	$1000	$900	$900	1725	$1,725,000	$1,552,500
	$1000	$800	$800	75	$75,000	$60,000
				Total	$2,075,000	$1,885,000

Reading Your Bill

Billing will still be the same except that you will see the split purchase if your purchases cross a pricing tier. Your bill will reflect a breakdown of costs similar to the example tiered price scenario discussed in the previous section. In the previous example, you save $390,000 (the difference between Total List Price Cost and the Total Paid Amount) using Reserved Instances pricing tiers.

Current Limitations

The following limitations currently apply:

- Reserved Instance ID: If your Reserved Instance purchase crosses into a discount pricing tier or crosses into a new discount pricing tier, your new purchase will go into a pending state while the Reserved Instance service processes the purchase. Processing involves purchasing two or more different Reserved Instances for you: An amount at either the undiscounted rate or the lower discounted rate, and an amount at either the discounted rate or the higher discounted rate.
- Describe Reserved Instances: The Reserved Instance ID returned by your purchase CLI command and API action will be different from the actual ID of the new Reserved Instances.

- Amazon EC2 Reserved Instance purchases are the only purchases that will apply toward your Amazon EC2 Reserved Instance pricing tiers. And the Amazon EC2 Reserved Instance pricing tiers apply only to Amazon EC2 Reserved Instance purchases.
- Amazon EC2 Reserved Instance pricing tiers do not apply to instances running Microsoft SQL Server.

Working with Reserved Instances

In this section, we walk you through the steps of finding, purchasing, and verifying your Reserved Instances.

AWS Management Console

To find and purchase a Reserved Instance

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Click **Reserved Instances** in the **Navigation** pane.
    The **Reserved Instances** page displays a list of your account's instances.
3. Click **Purchase Reserved Instances**.
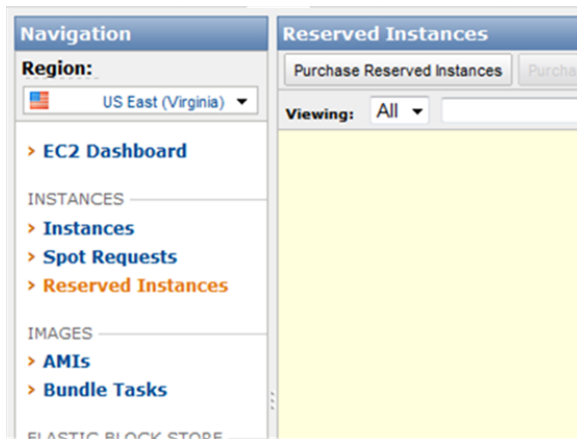
4. Locate a Reserved Instance to purchase by specifying the following:

    - Platform (if you want to use the Reserved Instance with Amazon Virtual Private Cloud, make sure to select a platform that includes *Amazon VPC* in its name)

    - Instance Type

    - Offering Type (For more information, go to Reserved Instances Offerings (p. 195).)

    - Availability Zone

    - Term

    - Tenancy (For more information, go to Dedicated Instances Basics in the *Amazon Virtual Private Cloud User Guide.*)

The **Purchase Reserved Instances** page displays the **One-time fee** and the **Hourly usage fee**.

5. Select the number of instances to purchase and click **Continue**.
   The **Purchase Reserved Instances** wizard prompts you to confirm your order.

6. To confirm your order, click **Purchase**.
   Your purchase is complete.

7. To verify your order, select **Reserved Instances**.

The **Reserved Instances** page displays a list of Reserved Instances that belong to your account.

You can run your Reserved Instance any time after your purchase is complete. To run your Reserved Instance, you launch an instance in the same way you launch an On-Demand instance. Make sure to specify the same criteria that you specified for your Reserved Instance. AWS will automatically charge you the lower hourly rate.

Command Line Tools

To find and purchase a Reserved Instance

1. Check to see which Reserved Instances are available.

```
PROMPT> ec2-describe-reserved-instances-offerings
```

Amazon EC2 returns output similar to the following:

```
PROMPT> ec2-describe-reserved-instances-offerings
ReservedInstancesOfferingId AvailabilityZone InstanceType Duration FixedPrice
 UsagePrice ProductDescription Currency InstanceTenancy OfferingType
OFFERING 248e7b75-c83a-48c1-bcf7-b7f03e9c43fe us-east-1b c1.medium 3y  700.0
 0.06 Linux/UNIX (Amazon VPC) USD default Medium Utilization
OFFERING 3a98bf7d-05c0-40d0-a173-81a3986ba568 us-east-1b c1.medium 3y  700.0
 0.125 Windows USD default Medium Utilization
OFFERING 4b2293b4-ff40-4a1a-9fef-1f12ad37a711 us-east-1b c1.medium 3y  700.0
 0.06 Linux/UNIX USD default Medium Utilization
OFFERING 60dcfab3-71e1-42cd-bd3e-f5e1e61ec5a8 us-east-1b c1.medium 3y  700.0
 0.125 Windows (Amazon VPC) USD default Medium Utilization
OFFERING 649fd0c8-cafc-4060-ace6-a074c9621f75 us-east-1b c1.medium 3y  865.0
 0.155 Windows (Amazon VPC) USD dedicated Medium Utilization
...
```

The preceding output shows a part of the overall offerings that are available. Here we show both 3-year (3y) and 1-year (1y) Reserved Instances available for m1.small instances in either the us-east-1b or us-east-1d Availability Zones.

> **Important**
>
> If you want to use the Reserved Instance with Amazon Virtual Private Cloud, make sure to select an offering that includes *Amazon VPC* in the description (e.g., Linux/UNIX (Amazon VPC)).

> **Tip**
>
> You can filter this list to return only certain types of Reserved Instances offerings of interest to you. For more information about how to filter the results, go to ec2-describe-reserved-instances-offerings in the *Amazon Elastic Compute Cloud Command Line Reference*.

2. From the list of available Reserved Instances, purchase the Reserved Instances that meet your requirements. To purchase a Reserved Instance, use `ec2-purchase-reserved-instances-offering`.

```
PROMPT> ec2-purchase-reserved-instances-offering --offering offering_id -
-instance-count count
```

Amazon EC2 returns output similar to the following:

```
PURCHASE   af9f760e-c1c1-449b-8128-1342d3a6927a   438012d3-80c7-42c6-9396-
a209c58607f9
```

The response includes the offering ID and a reservation ID.

3.   Write down or save the reservation ID for future reference.

4.   Verify the purchase:

```
PROMPT> ec2-describe-reserved-instances
```

Amazon EC2 returns output similar to the following:

```
RESERVEDINSTANCE   af9f760e-c1c1-449b-8128-1342d3a6927a    us-east-1b
m1.small        1y      227.5   0.03    Linux/UNIX  Active
```

You can run your Reserved Instance any time after your purchase is complete. To run your Reserved Instance, you launch an instance in the same way you launch an On-Demand instance. Make sure to specify the same criteria that you specified for your Reserved Instance. AWS will automatically charge you the lower hourly rate.

API

To find and purchase a Reserved Instance

1.   Check to see which Reserved Instances are available and where they are located using the following Query request.

```
https://ec2.amazonaws.com/
?Action=DescribeReservedInstancesOfferings
&...auth parameters...
```

Following is an example response (note that the duration is in seconds; 31536000 seconds equals 1 year).

```
<DescribeReservedInstancesOfferings xmlns="http://ec2.amazonaws.com/doc/2012-
07-20/">
  <reservedInstancesOfferingsSet>
    <item>
      <reservedInstancesOfferingId>438012d3-80c7-42c6-9396-
a209c58607f9</reservedInstancesOfferingId>
      <instanceType>m1.small</instanceType>
      <availabilityZone>us-east-1b</availabilityZone>
      <duration>31536000</duration>
      <fixedPrice>227.5</fixedPrice>
      <usagePrice>0.03</usagePrice>
      <productDescription>Linux/UNIX</productDescription>
    </item>
    ...
  </reservedInstancesOfferingsSet>
</DescribeReservedInstancesOfferings>
```

Important

If you want to use the Reserved Instance with Amazon Virtual Private Cloud, make sure to select an offering that includes *Amazon VPC* in the product description (e.g., Linux/UNIX (Amazon VPC)).

Tip

You can filter this list to return only certain types of Reserved Instances offerings of interest to you. For more information about how to filter the results, go to DescribeReservedInstancesOfferings in the *Amazon Elastic Compute Cloud API Reference*.

2. From the list of available Reserved Instances, purchase an offering using a Query request similar to the following.

```
https://ec2.amazonaws.com/
?Action=PurchaseReservedInstancesOffering
&ReservedInstancesOfferingId=438012d3-80c7-42c6-9396-a209c58607f9
&instanceCount=2
&...auth parameters...
```

Following is an example response.

```
<PurchaseReservedInstancesOffering xmlns="http://ec2.amazonaws.com/doc/2012-
07-20/">
   <reservedInstancesId>af9f760e-c1c1-449b-8128-1342d3a6927a</reservedIn
stancesId>
</PurchaseReservedInstancesOffering>
```

3. To verify the purchase, construct the following Query request.

```
https://ec2.amazonaws.com/
?Action=DescribeReservedInstances
&...auth parameters...
```

Following is an example response (note that the duration is in seconds; 31536000 seconds equals 1 year):

```
<DescribeReservedInstances xmlns="http://ec2.amazonaws.com/doc/2012-07-20/">

   <requestId>59dbff89-35bd-4eac-99ed-be587EXAMPLE</requestId>
   <reservedInstancesSet>
     <item>
      <reservedInstancesId>af9f760e-c1c1-449b-8128-1342d3a6927a</reservedIn
stancesId>
        <instanceType>m1.small</instanceType>
        <availabilityZone>us-east-1b</availabilityZone>
        <duration>31536000</duration>
        <usagePrice>227.5</usagePrice>
        <fixedPrice>0.03</fixedPrice>
        <instanceCount>2</instanceCount>
        <productDescription>Linux/UNIX</productDescription>
        <state>Active</state>
     </item>
   </reservedInstancesSet>
</DescribeReservedInstances>
```

You can run your Reserved Instance any time after your purchase is complete. To run your Reserved Instance, you launch an instance in the same way you launch an On-Demand instance. Make sure to

specify the same criteria that you specified for your Reserved Instance. AWS will automatically charge you the lower hourly rate.

Related Topics

- Reserved Instances FAQs (p. 552)

Launching Amazon EC2 Instances

Topics

- Launching an Instance from an AMI (p. 217)
- Launching an Instance from a Snapshot (p. 222)
- Launching an Instance from a Virtual Machine (p. 224)

This section describes how to launch Amazon EC2 instances.

Related Topics

- Amazon Machine Images (AMI) (p. 23)

- Amazon EC2 Instances (p. 84)

- Region and Availability Zone Concepts (p. 111)

Launching an Instance from an AMI

This topic describes how to launch and run an instance from an AMI.

Important

After launching an instance, you are billed hourly for running time. When you are finished with this task, be sure to terminate any instances that you started. For more information, see Terminating Instances (p. 314).

If the instance's state immediately goes to "terminated" instead of "running", you can get information about why the instance didn't launch. For more information, see What to Do If an Instance Immediately Terminates (p. 346).

AWS Management Console

You can either leverage the Free Usage Tier to launch and use a free Amazon EC2 Micro Instance for 12 months, or you can launch a larger instance type, but not within the Free Usage Tier. For more information about the Free Usage Tier, go to the AWS Free Usage Tier product page and Getting Started with AWS Free Usage Tier.

If you launch an instance that is not within the Free Usage Tier, you incur the standard Amazon Elastic Compute Cloud usage fees for the instance. For more information about Amazon EC2 usage rates, go to the Amazon EC2 product page.

Important

If you launch an instance that is not within the Free Usage Tier, you are billed after you launch the instance and charged for the time that the instance is running even if it remains idle.

To launch an instance

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. From the Amazon EC2 console dashboard, click **Launch Instance**.

   The **Create a New Instance** page provides the following ways to launch an instance:

- The **Classic Wizard** provides more granular control and advanced settings for configuring your instance.

- The **Quick Launch Wizard** simplifies the process for you by automatically configuring many selections to get you started quickly with your instance. If you are just trying out Amazon EC2, see Getting Started with Amazon EC2 Linux Instances (p. 13) to get started using the Quick Launch Wizard.

- The **AWS Marketplace** is an online store where you can buy software that runs on AWS. You can launch your instance with just one click.

   For this procedure, we'll use the Classic Wizard. Click **Classic Wizard**, and then click **Continue**.

3.   On the **CHOOSE AN AMI** page, the **Quick Start** tab displays a list of basic configurations called Amazon Machine Images (AMIs). An AMI contains all the information needed to create a new instance of a server. For example, an AMI might contain the software required to act as a web server (for example, Linux, Apache, and your website). To keep things simple, AWS marks the AMIs that are available in the Free Usage Tier with a star.

   As you are selecting an AMI, it's important to note whether the AMI is backed by instance store or by Amazon EBS. Select the type of AMI that meets your needs. For more information, see Storage for the Root Device (p. 24).

   Choose the AMI that you want to use and click its **Select** button.

4.   On the **INSTANCE DETAILS** page, change the following settings as necessary, and then click **Continue**.

- **Number of Instances**: Select the number of instances based on the selected AMI that you want to launch.

- **Instance Type**: Select the size of the instance you want to launch. Larger instance types have more CPU and memory.

- **Launch as an EBS-Optimized Instance**: If the instance type supports this feature, select this check box to enable it. (Otherwise, this option is grayed out.) Additional charges apply. For more information, see EBS-Optimized Instances (p. 88).

- **Availability Zone**: Select the availability zone that you want. We recommend that you choose `No Preference`.

5.   On the second **INSTANCE DETAILS** page, change the following advanced instance options as necessary, and then click **Continue**.

- **Kernel ID**: Select **Use Default** unless you want to use a specific kernel. For more information, see Kernels and RAM Disk FAQ.

- **RAM Disk ID**: Select **Use Default** unless you want to use a specific RAM disk. For more information, see Kernels and RAM Disk FAQ.

- **Monitoring**: Select this check box to enable detailed monitoring of your instance using Amazon CloudWatch. Additional charges will apply. For more information about Amazon CloudWatch, see Introduction to Amazon CloudWatch in the *Amazon CloudWatch Developers Guide*.

- **User Data**: You can specify user data to configure an instance during launch or to run a configuration script.

- **base64 encoded**: Select this check box if you want data to be base64 encoded.

- **Termination Protection**: Select this check box to prevent against accidental termination. For more information, see Enabling Termination Protection for an Instance (p. 315).

- **Shutdown Behavior**: Select whether the instance should stop or terminate when shut down.

- **IAM Role**: Select an AWS Identity and Access Management (IAM) role to associate with the instance. For more information, see Using IAM Roles with Amazon EC2 Instances (p. 361).

6. On the third **INSTANCE DETAILS** page, you can click **Edit** to specify volumes to attach to the instance besides the volumes specified by the AMI (such as the root device volume), and then click **Continue**. For this example, it's fine to stick with the default Root volume. For more information about changing the block device mapping, see Updating the Block Device Mapping when Launching an Instance (p. 495).

7. On the final **INSTANCE DETAILS** page, specify tags for the instance, and then click **Continue**. For more information about tags, see Tagging Your Resources (p. 508).

8. On the **CREATE KEY PAIR** page, you can choose from any existing key pairs that you have created or create a new one. For this example, we'll create a key pair.

   Important

   Do not select the **None** option. If you launch an instance without a key pair, you can't connect to it. This option is used only when you are creating your own AMI and don't need to connect to the instance.

   a. Click **Create a new Key Pair**.
   b. Type a name for your key pair and then click **Create & Download your Key Pair**. You need the contents of the private key to connect to your instance after it is launched. Amazon Web Services does not keep the private portion of key pairs.
   c. Save the private key in a safe place on your system. Note the location because you'll need the key to connect to the instance.

9. On the **CONFIGURE FIREWALL** page, the wizard automatically selects a security group for you.

   A security group defines firewall rules for your instances. These rules specify which incoming network traffic is delivered to your instance. All other traffic is ignored.

   If you're new to Amazon EC2 and haven't set up any security groups yet, AWS defines a default security group for you. If you use an Amazon Machine Image (AMI) on the **Quick Start** tab, the name and description for the group is quick-start-*x* where *x* is a number associated with your quick-start group. The first security group you create using the **Quick Start** tab is named quick-start-1. You can change the name and description. For an Amazon Linux instance, you connect through SSH on port 22. The quick-start-*x* security group automatically allows SSH traffic on port 22. For a Windows instance, you connect through Remote Desktop Protocol (RDP) and the security group automatically allows RDP traffic on port 3389.

   If you have used Amazon EC2 before, the wizard looks for an existing security group for the type of instance you're creating.

   Caution

   The quick-start-*x* security group enables all IP addresses to access your instance over the specified ports (i.e., SSH for Linux/UNIX or RDP for Windows). This is acceptable for the short exercise in this tutorial, but it's unsafe for production environments. In production, you'll authorize only a specific IP address or range of addresses to access your instance.

   Note

   After an instance is running, you can't change which security groups it belongs to, but you can add and remove rules.

10. Click **Continue** to review your settings. When you're satisfied with your selections, click **Launch** to begin launching your instance.

Command Line Tools

To launch an instance

1. Use the ec2-run-instances command.

```
PROMPT>  ec2-run-instances ami-3ac33653 -k gsg-keypair
```

Amazon EC2 returns output similar to the following example.

```
RESERVATION      r-f25e6f9a      111122223333    default
INSTANCE    i-85b435ee    ami-3ac33653    pending    gsg-keypair    0
m1.small    2010-03-30T08:01:36+0000    us-east-1a    aki-407d9529    monit
oring-disabled    ebs
```

2. Look for the instance ID in the second field and write it down.
   You use it to manipulate this instance (including terminating it when you are finished).

   It takes a few minutes for the instance to launch.

3. The ec2-describe-instances command displays the launch status of the instance.

```
PROMPT>  ec2-describe-instances i-85b435ee
RESERVATION r-f25e6f9a 111122223333    default
INSTANCE i-85b435ee ami-3ac33653 ec2-67-202-28-13.compute-1.amazonaws.com
domU-12-31-39-00-78-93.compute-1.internal running gsg-keypair 0 m1.small
2010-03-30T08:01:36+0000 us-east-1a aki-407d9529 monitoring-disabled
67.202.28.13 10.254.127.97 ebs
BLOCKDEVICE /dev/sda1 vol-02a2a46b 2010-03-30T08:01:44.000Z
```

When the instance state in the field just before the key pair name reads "running," the instance has started booting. There might be a short time before it is accessible over the network, however. The first DNS name is your instance's external DNS name, i.e. the one that can be used to contact it from the Internet. The second DNS name is your instance's local DNS name, and can only be contacted by other instances within the Amazon EC2 network. The DNS names of your instances are different than those shown in the preceding example and you should use yours instead. The examples in this guide use the public DNS name.

API

To launch an instance

1. Use the RunInstances action. Construct the following request.

```
https://ec2.amazonaws.com/
?Action=RunInstances
&ImageId=ami-3ac33653
&MaxCount=1
&MinCount=1
```

```
&KeyName=gsg-keypair
&Placement.AvailabilityZone=us-east-1a
&AUTHPARAMS
```

The following is an example response.

```
<RunInstancesResponse xmlns="http://ec2.amazonaws.com/doc/2012-07-20/">
  <reservationId>r-47a5402e</reservationId>
  <ownerId>111122223333</ownerId>
  <groupSet>
    <item>
      <groupId>default</groupId>
    </item>
  </groupSet>
  <instancesSet>
    <item>
      <instanceId>i-2ba64342</instanceId>
      <imageId>ami-3ac33653</imageId>
      <instanceState>
        <code>0</code>
        <name>pending</name>
      </instanceState>
      <privateDnsName/>
      <dnsName/>
      <reason/>
      <amiLaunchIndex>0</amiLaunchIndex>
      <instanceType>m1.small</instanceType>
      <launchTime>2007-08-07T11:51:50.000Z</launchTime>
      <placement>
        <availabilityZone>us-east-1a</availabilityZone>
        <groupName/>
        <tenancy>default</tenancy>
      </placement>
      <monitoring>
        <state>disabled</state>
      </monitoring>
      <rootDeviceType>ebs</rootDeviceType>
    </item>
  </instancesSet>
</RunInstancesResponse>
```

2. Look for the instance ID in the `instanceId` element in the response and write it down.

    You use it to manipulate this instance (including terminating it when you are finished).

    It takes a few minutes for the instance to launch.

3. The DescribeInstances action displays the launch status of the instance.

    Construct the following request.

```
https://ec2.amazonaws.com/
?Action=DescribeInstances
&InstanceId=i-2ba64342
&AUTHPARAMS
```

The following is an example response.

```
<DescribeInstancesResponse xmlns="http://ec2.amazonaws.com/doc/2012-07-20/">

  <reservationId>r-47a5402e</reservationId>
  <ownerId>111122223333</ownerId>
  <groupSet>
    <item>
      <groupId>default</groupId>
    </item>
  </groupSet>
  <instancesSet>
    <item>
      <instanceId>i-2ba64342</instanceId>
      <imageId>ami-3ac33653</imageId>
      <instanceState>
        <code>16</code>
        <name>running</name>
      </instanceState>
      <privateDnsName></privateDnsName>
      <dnsName></dnsName>
      <keyName>gsg-keypair</keyName>
      <amiLaunchIndex>0</amiLaunchIndex>
      <instanceType>m1.small</instanceType>
      <launchTime>2007-08-07T11:51:50.000Z</launchTime>
      <placement>
        <availabilityZone>us-east-1a</availabilityZone>
        <groupName/>
        <tenancy>default</tenancy>
      </placement>
      <monitoring>
        <state>disabled</state>
      </monitoring>
    </item>
  </instancesSet>
</DescribeInstancesResponse>
```

When the instance state is `running`, as it is in the previous response, the instance has started booting.

Launching an Instance from a Snapshot

If you have a snapshot of the root device volume of an instance, you can terminate that instance and then later launch a new instance from the snapshot. This is useful if you don't have the original AMI available to launch new instances from.

To launch a new instance from the snapshot, you register the snapshot and then launch the resulting AMI.

Important

Registering a snapshot works only for Linux/UNIX AMIs; although you can register a snapshot to create a Windows AMI, the AMI isn't launchable.

Command Line tools

1. Use the ec2-register command and specify a block device mapping that maps the root device name of your choice to the snapshot.

The following example specifies the root device as /dev/sda1, and maps it to the snap-12345678 snapshot. The resulting root device volume is the same size as the snapshot, and it will automatically be deleted on instance termination. If you were to specify the block device mapping as `/dev/sda1=snap-1234578::false`, then the volume would persist on instance termination.

```
PROMPT>  ec2-register -n My_Image_Name -d My_image_description --root-device-
name /dev/sda1 -b /dev/sda1=snap-12345678 -aki-12345678
```

The response displays the ID for your new AMI.

```
IMAGE    ami-61a54008
```

The AMI now appears in the list of AMIs that you own. You can view that list in the AWS Management Console, or by using the following command: `ec2-describe-images -o self`.

2. Launch an instance of the AMI.

The resulting instance has a root device volume created from the snapshot.

API

1. Issue the following Query request to register an image.

The example specifies the root device as /dev/sda1, and maps it to the snap-12345678 snapshot. The resulting root device volume is the same size as the snapshot, and it will automatically be deleted on instance termination. You can set `DeleteOnTermination` to `false` if you'd rather the volume persist.

```
https://ec2.amazonaws.com/
?Action=RegisterImage
&Name=MyImage
&KernelId=aki-f70657b2
&RamdiskId=ari-ff0d5cba
&RootDeviceName=/dev/sda1
&BlockDeviceMapping.1.DeviceName=/dev/sda1
&BlockDeviceMapping.1.Ebs.SnapshotId=snap-12345678
&AUTHPARAMS
```

For information about the auth parameters, go to Common Query Parameters in the *Amazon Elastic Compute Cloud API Reference*.

The following is an example response.

```
<RegisterImageResponse xmlns="http://ec2.amazonaws.com/doc/2012-07-20/">
   <requestId>59dbff89-35bd-4eac-99ed-be587EXAMPLE</requestId>
   <imageId>ami-61a54008</imageId>
</RegisterImageResponse>
```

The AMI now appears in the list of AMIs that you own. You can view that list by using `DescribeImages` with *Owner*=self.

2. Launch an instance of the AMI.

The resulting instance has a root device volume created from the snapshot.

Launching an Instance from a Virtual Machine

Topics

Using Instances of Your Virtual Machine in Amazon EC2

Topics

There are two ways you can launch an instance in the Amazon Elastic Compute Cloud (Amazon EC2). You can launch an instance from an AMI that you created or selected from a catalog. Or, you can launch an instance from a virtual machine (VM) that you imported from a Citrix Xen, Microsoft Hyper-V, or VMware vSphere virtualization platform. This section covers using VMs from Citrix, Microsoft, or VMware to launch instances.

To use your virtual machine as an instance in Amazon EC2, you must first export it from the virtualization platform using the platform's tools. Then you import it to Amazon EC2 using the Amazon EC2 command line or API tools.

If you are importing a VMware vSphere VM, you can also use the Amazon EC2 VM Import Connector for VMware (Connector), a plug-in that integrates with VMware vSphere Client, to perform the task.

Whether you use the command line tools or the API, or the Connector, you will follow the same general process for importing VMs or volumes to Amazon EC2. You need to complete these tasks, which are all discussed in this section:

1. Prepare the virtual machine for import to Amazon EC2. For more information, go to Before You Get Started (p. 225)
2. Export the virtual machine from the external environment.
   - Exporting from Citrix (p. 241)
   - Exporting from Microsoft Hyper-V (p. 244)
   - Exporting from VMware (p. 246)
3. Import the virtual machine to Amazon EC2.
   For information about using the command line tools to import your VM, see Using the Command Line Tools to Import Your Virtual Machine to Amazon EC2 (p. 240). For information about using the Connector, see Using the Amazon EC2 VM Import Connector to Import Your Virtual Machine to Amazon EC2 (p. 226).
4. Upload the instance to Amazon EC2.
5. Launch the instance in Amazon EC2.

The following figure shows the process of importing a VM or volume.
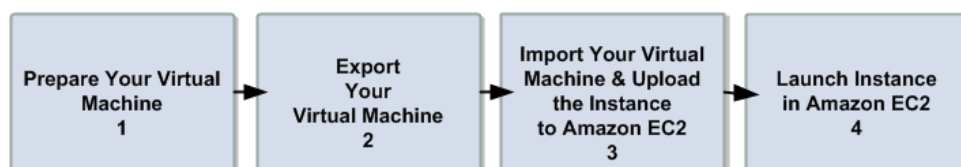
Components in Your VM Environment

The table in this section describes the typical components in your VM environment.

Component	Description	Product Name
Virtualization product	Virtualization service for managing virtual computing infrastructure	Citrix Xen Microsoft Hyper-V VMware vSphere (vSphere)
Client	The software you need on your computer to access and manage your virtualization platform environment	Citrix Xen Center Microsoft Hyper-V Manager VMware vSphere Client
Server	The management platform for the virtualization environment	Citrix XenServer Microsoft Hyper-V VMware vCenter Server
Amazon EC2 VM Import Connector for VMware (Connector)	The virtual appliance, a plug-in to the management platform of the virtualization Server, that enables the import of virtual machines into Amazon EC2 using the Client interface	Citrix Xen (not applicable) Microsoft Hyper-V (not applicable) VMware vCenter—Amazon EC2 VM Import Connector for VMware vCenter (Connector)

Before You Get Started

This section discusses the things you need to know and what you must have before you begin the process of importing your virtual machine.

Operating Systems—The following operating systems can be imported to Amazon EC2:

- Microsoft Windows Server 2003 R2 (Standard, Datacenter, Enterprise).
- Microsoft Windows Server 2008 R1 and R2 (Standard, Datacenter, Enterprise).

Image Formats—We support import of the following image formats for importing both volumes and instances to Amazon Web Services:

- RAW format for importing volumes and instances.
- Virtual Hard Disk (VHD) image formats, which are compatible with Microsoft Hyper-V and Citrix Xen virtualization products.
- ESX Virtual Machine Disk (VMDK) image formats, which are compatible with VMware ESX and VMware vSphere virtualization products.

> **Note**
>
> VMDK images from VMware Workstation are not compatible.

Known Limitations—The importing of instances and volumes is subject to the following limitations:

- You can have up to five conversions and tasks in progress at the same time per Region.
- Typically, you import a compressed version of a disk image; the expanded image cannot exceed 1TB.

- Tasks must complete within 7 days of the start date.
- Importing virtual machines with more than one virtual disk is not supported. We suggest that you import the VM with only the boot volume, and import any additional disks using `ImportVolume` (`ec2-import-volume`) in the command line. After the `ImportInstance` task is complete, use `AttachVolume` (`ec2-attach-volume`) to associate the additional volumes with your instance.
- Multiple network interfaces are not currently supported. When converted and imported, your instance will have a single virtual NIC using DHCP for address assignment.
- For vCenter 4.0 and vSphere 4.0 users, remove any attached CD-ROM images or ISOs from the virtual machine.

Preparing Your Virtual Machine

Use the following guidelines to configure your virtual machine before exporting it from the virtualization environment.

> **Important**
>
> If you are importing a virtual machine from Citrix Xen, you must uninstall the Citrix Tools for Virtual Machines from the VM. If you don't uninstall the tools, your import will fail. For more information, see Exporting from Citrix (p. 241).

- Enable Remote Desktop (RDP) for remote access.
- Make sure your host firewall (Windows firewall), if configured, allows access to RDP. Otherwise, you will not be able to access your instance after the conversion is complete.
- Make sure all user accounts use secure passwords. This includes the administrator account.

> **Note**
>
> All accounts must have passwords. Otherwise, the import might fail.

- Disable any antivirus or intrusion detection software on your virtual machine. These services can be re-enabled after the import process is complete.
- Disconnect any CD-ROM drives (virtual or physical).

Using the Amazon EC2 VM Import Connector to Import Your Virtual Machine to Amazon EC2

Topics

You can use the Amazon EC2 VM Import Connector virtual appliance (vApp), a plug-in for VMware vCenter, to import virtual machines from your VMware vSphere infrastructure to Amazon EC2. The Connector is a virtual appliance that works with VMware vCenter Server only. It provides an easy-to-use interface, enhancing your existing management tools to work with the Amazon EC2 VM Import Connector.

> **Note**
>
> You cannot use the Connector to import Citrix Xen or Microsoft Hyper-V virtual machines to Amazon EC2. Instead, use the command line tools to import your Citrix and Hyper-V virtual machines to Amazon EC2. You can also choose to use the command line tools to import your

VMware VMs. For more information, see Using the Command Line Tools to Import Your Virtual Machine to Amazon EC2 (p. 240).

Using the Connector simplifies the process of importing your VMware VMs: You prepare the virtual machine for import to Amazon EC2, and then you import it to EC2.

The following figure shows the process of importing VMware VMs using the Connector and without using the Connector.
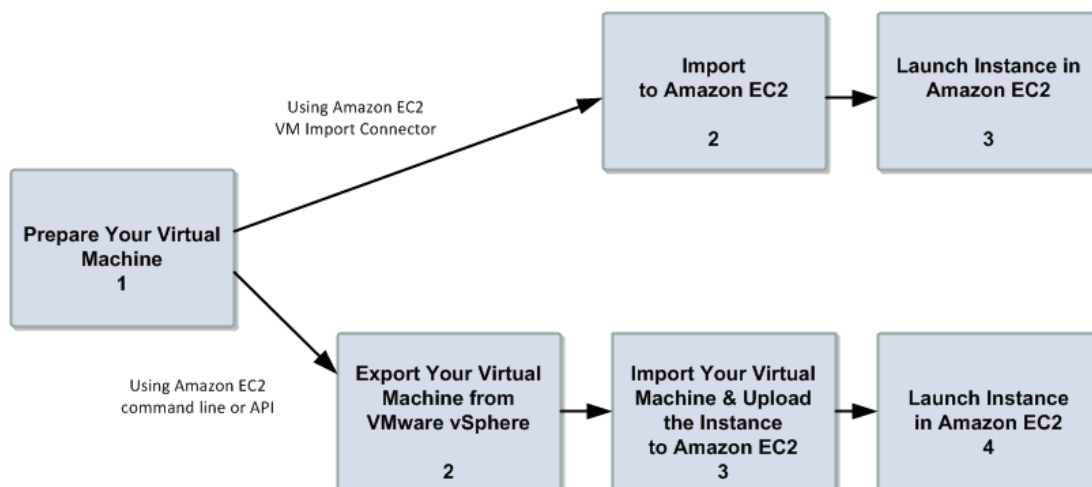
Before You Install the Connector

To use the VM Import Connector, you first need to install the Connector virtual appliance. Before you install, read through the general prerequisites listed in the Before You Get Started (p. 225) section and make sure that your environment meets the following requirements:

VMware infrastructure

- vSphere 4.0 or 4.1
- vCenter 4.0 or 4.1

Amazon EC2 VM Import Connector vApp

- 256MB RAM
- Minimum 250GB of disk space

> **Note**
>
> Although the Connector virtual appliance is small, it temporarily stores the VM images that you import to Amazon EC2. The data store must be large enough to accommodate these images, plus the Connector. We recommend a data store size of 250GB or larger.
>
> To estimate the disk space you need, multiply the maximum number of parallel import tasks you want to run with the Connector by the average size you expect your VMs to be, then add about 10GB for the virtual appliance. For example, if you project that you will run a maximum of 5 imported VMs averaging 75GB in size, then you'll need about 385GB of disk space.

Internet access

- Outbound Internet access, either direct or via a proxy, from the Connector appliance on TCP port 443 (SSL) to Amazon EC2 and Amazon S3.

Note

If you are adding a firewall rule to allow this access and want to further restrict this access to Amazon EC2 and Amazon S3, you can add the hosts at the published endpoints as the destinations on TCP port 443. For more information about the current endpoints go to Regions and Endpoints.

- DHCP server
- A static IP address or an IP reservation via DHCP for the virtual appliance

Note

You must set up a static DHCP lease or configure a static IP before you begin the configuration process.

Connector local network access

- Inbound TCP 443 (SSL) from vCenter Server and vSphere Client
- Inbound TCP 80 (HTTP) from the LAN for Connector Web Console

Administrative rights

- To VMware vSphere and VMware vCenter for installation. For information on how to allow a user without administrative rights to use the Connector to import VMs to Amazon EC2, see To grant permission to non-administrative users to import VMs to Amazon EC2 (p. 234).

AWS access credentials

- The Connector stores AWS credentials for each VMware vCenter user. In this way, multiple users with separate AWS credentials can use the same Connector. Alternatively, you can use the same AWS account and credentials with more than one user. Each VMware vSphere user will need to fill out the information in the **Enter AWS Credentials** dialog box the first time he or she uses the Connector. For information on how to get your AWS credentials, see How to Get Your Access Key ID and Secret Access Key (p. 352).
- Your AWS account must be subscribed to Amazon EC2 before you start the import process.

Note

The Connector virtual appliance stores your AWS credentials. To protect these credentials from unauthorized use, grant access to the virtual appliance's console and configuration only to administrators.

Installing the Connector for VMware vCenter

After you have confirmed that you have all the prerequisites and your environment meets the minimum requirements, you are now ready to install the Connector virtual appliance (vApp) for VMware vCenter.

The Connector virtual appliance is an Open Virtualization Format (OVF) package that is distributed in an Open Virtualization Application (OVA) file. Installing the Connector involves downloading the OVA file and deploying the OVF template.

To Install the Connector for the VMware vCenter

1. Download the OVA package for the Connector virtual appliance from Amazon Web Services Developer Tools and save it to your Downloads folder.

2. Start the vSphere Client and connect to your vCenter Server.

3. Using the vSphere Client, deploy the OVF template contained in the OVA file that you downloaded.
   On the **File** menu, select **Deploy OVF Template** and point to the location where you downloaded
   the OVA package.
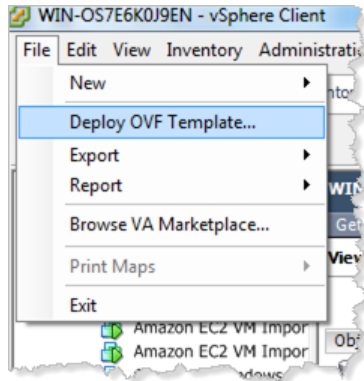
A series of **Deploy OVF Template** screens walks you through the deployment process.

4. Confirm that the **Deploy OVF Template** screen is displaying correct information about the VM Import
   Connector, and click **Next**.
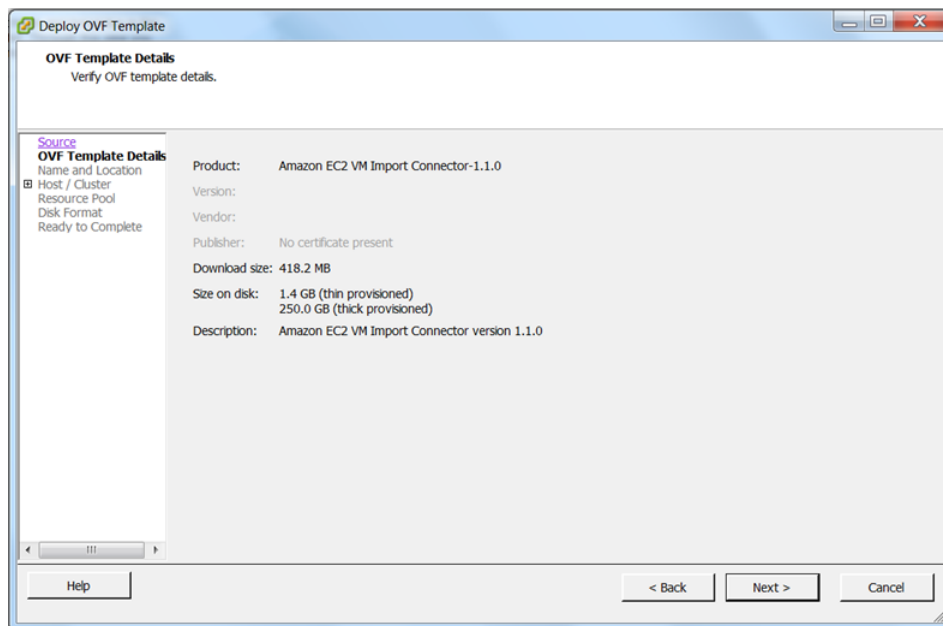
5. Specify the name and location of the Connector or accept the default, then click **Next**.
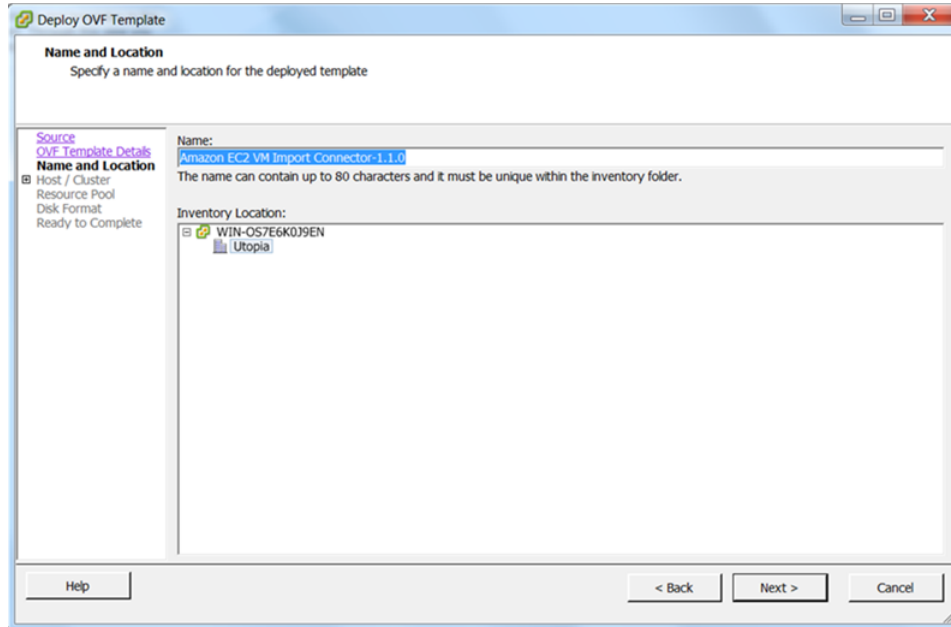
6. Select the host or cluster in which you want the Connector to run, then click **Next**.
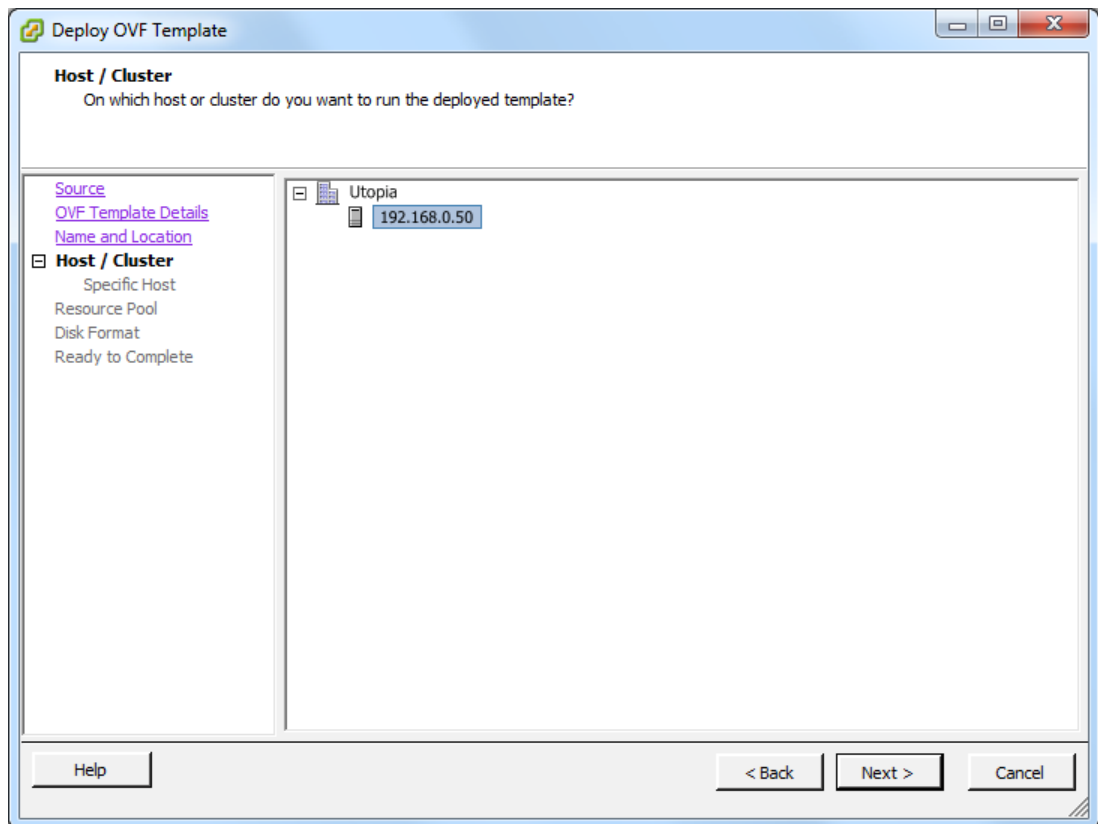
7. Select the data store where the Connector will be stored, then click **Next**.

Note

Although the Connector virtual appliance is small, it temporarily stores the images of the
virtual machines that import to Amazon EC2. Therefore, the data store must be large enough
to accommodate these images, as well as the Connector. We recommend a data store size
of 250GB or larger.
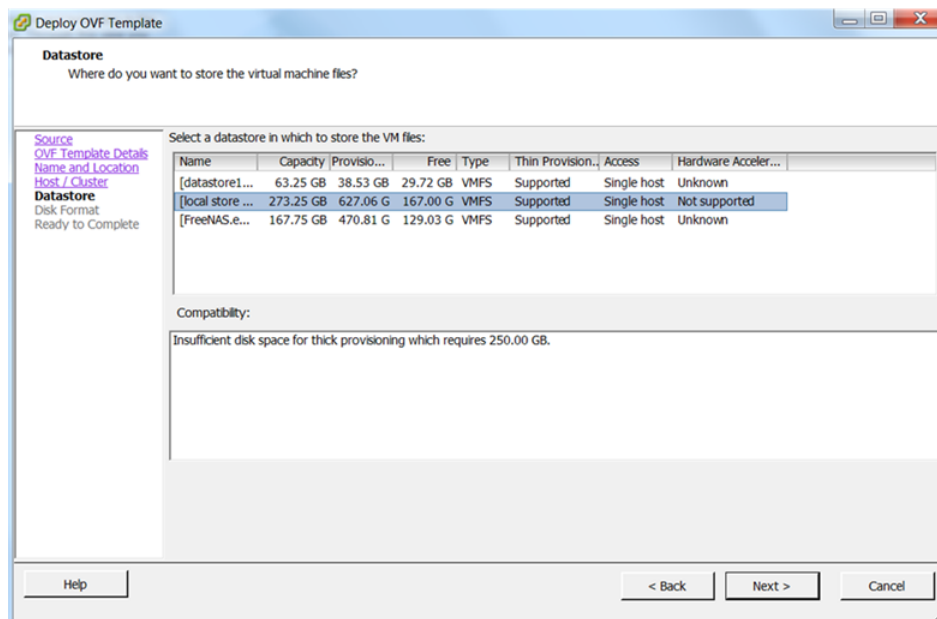
8.   Select **Thin provisioned format**, then click **Next**.
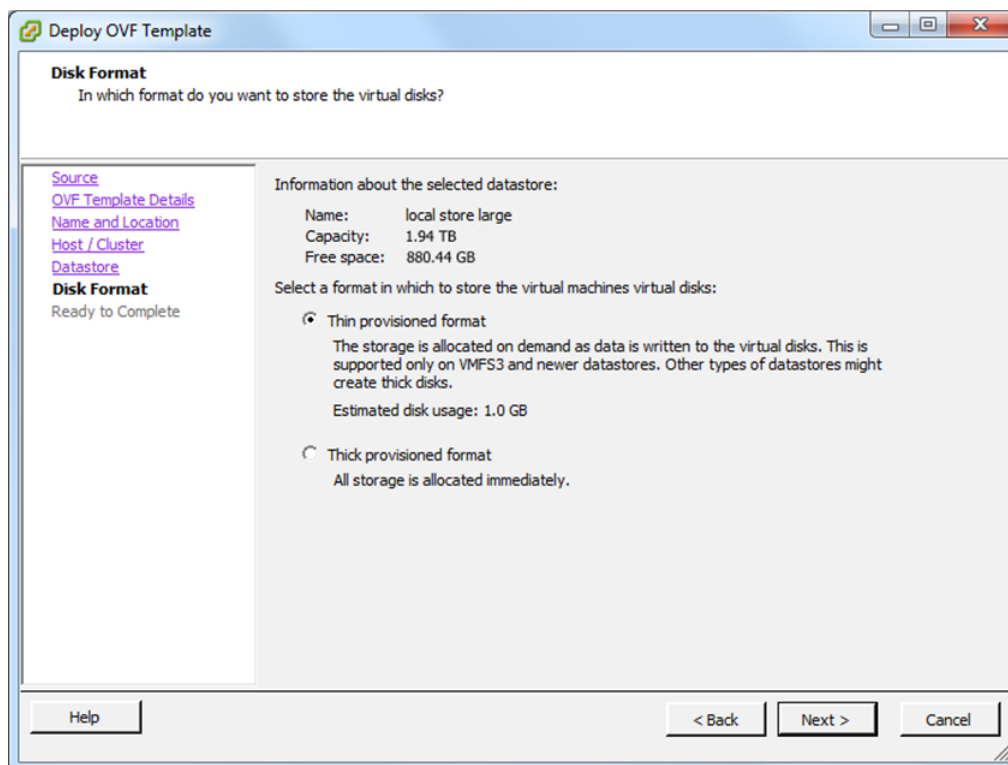
9.   Confirm the details you selected and click **Finish**.

The Connector virtual appliance will now install.

Configuring the Connector for VMware vCenter

After you install the EC2 VM Import Connector, you must obtain its IP address and password and register it with the vCenter Server. To obtain the Connector's IP address and password, you first start the Connector appliance in vCenter, then go to the vCenter **Console** tab. The tab displays the IP address and password when the Connector is running. In a web browser, use this information to log in to the Connector and register it with the vCenter Server.

If the VMware vSphere Client was running when you installed the Connector virtual application, close and restart the vSphere Client, then follow these procedures.

To start the Connector for VMware vCenter

1.   On the vSphere Client, right-click the Connector you just installed, select **Power** and then **Power On**.
2.   To open the console, right-click the Connector you just installed and click **Open Console**.

When the Connector is running, you will find its IP address and password displayed in the vCenter **Console**. Your Connector information will be similar to the following example:

To register the Connector with the VMware vCenter

1.   Open a web browser and, in the address bar, type the Connector IP address that you obtained from the **Console** in the vCenter, and log in with the password.
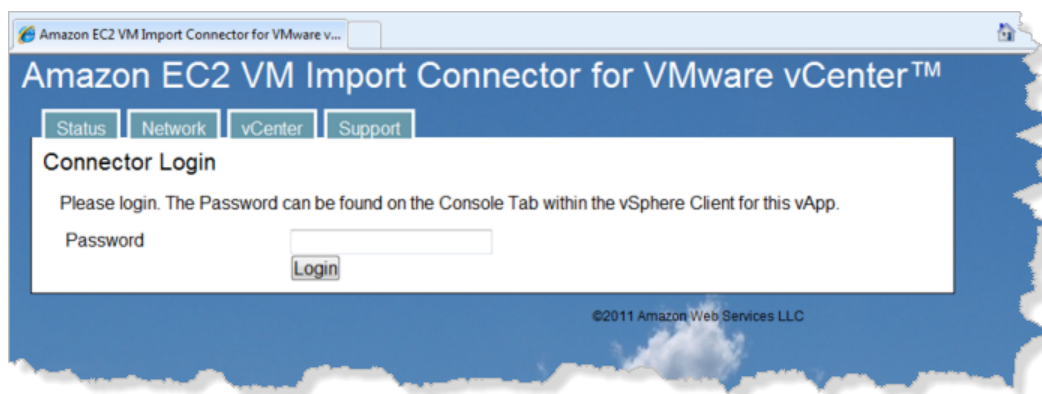
2.   When you are logged in, the browser displays Connector status information. Note that the Connector is not yet registered with the vCenter Server. Confirm that everything else in the **Connector Status** list has a status of **OK**.

3. If the version of the Connector that you are registering is not an upgrade, click **Register Now**. If you are upgrading the Connector, you must take the next two steps before registering.

   • Confirm that all import tasks are complete.
   • Deregister the old Connector from vCenter. To do this, go to the **vCenter** tab and click **Deregister**.
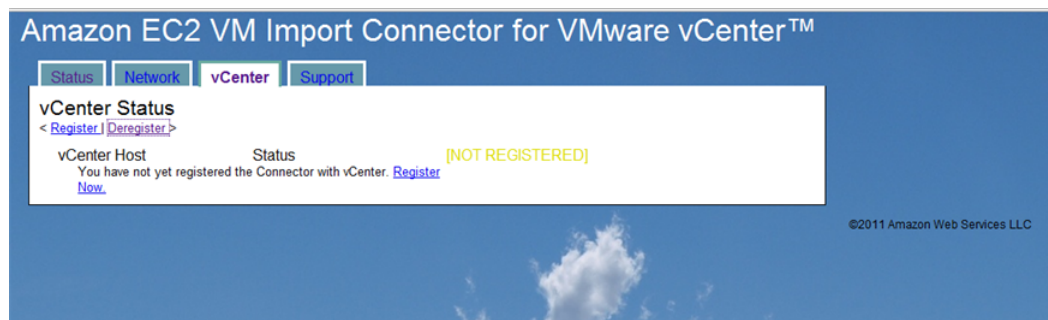
   The **vCenter Connector Registration** page appears.

4. Provide the IP address of the vCenter with which you want to register the Connector. The user name and password you use must have administrative rights. Click **Register Connector with vCenter**.

   Note

   If an error occurs, check that the VMware vCenter IP address or name that you provided is correct. Also, confirm that the Connector virtual appliance has network access to TCP port 443 on your vCenter Server. If the user name or password you provided is incorrect, you will see a description of this error.

To configure a proxy

If you need to configure a proxy to allow the Connector to reach the Internet, you can do it using the Connector's web interface.

1. Go to the **Network** tab, click **Configure Proxy**.
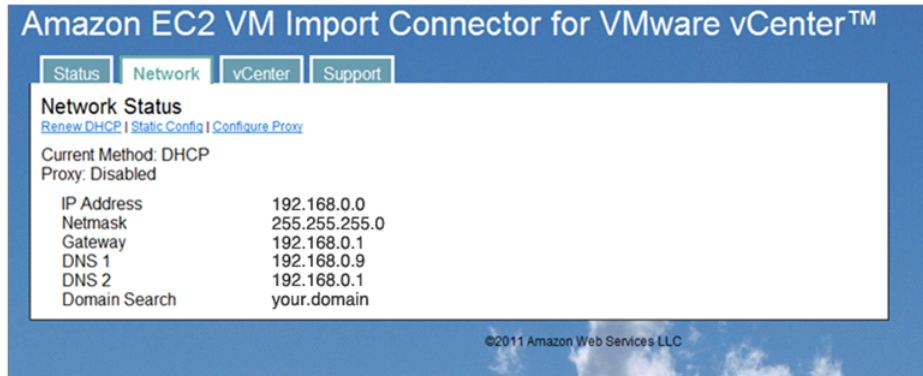
The **Network Configuration:Proxy** page appears.

2. Provide the information required and click **Configure Proxy**.
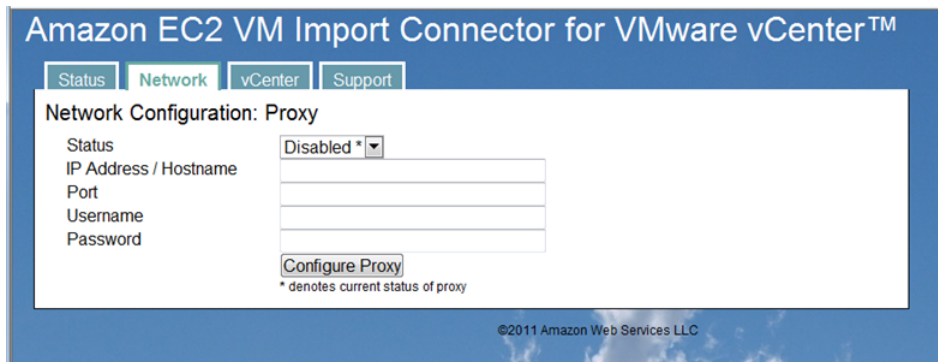
After using the web browser to register the Connector with the vCenter Server, as an administrator you can import virtual machines to Amazon EC2. If you're going to import VMs using only an administrator account, skip the following section and go to Using the Connector for VMware vCenter (p. 237). If you want non-administrative users to import VMs to Amazon EC2, you must grant them permission using the vCenter.

To grant permission to non-administrative users to import VMs to Amazon EC2

1. Log in to vCenter as Administrator and from **Home**, navigate to **Roles**, and click **Add Role**.
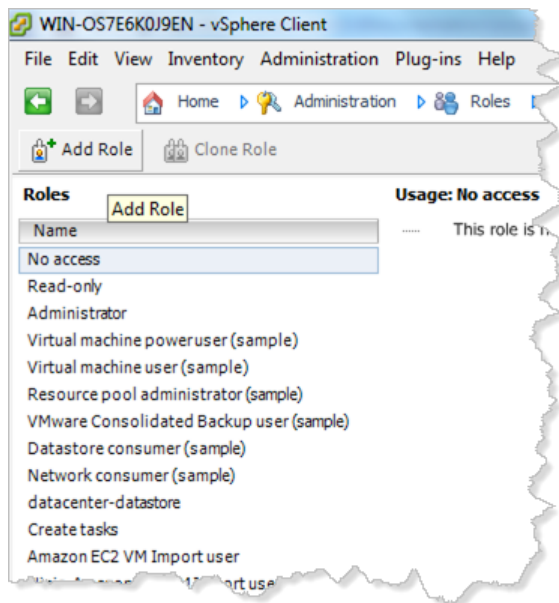
2. In the **Add New Role** dialog box, type the name for the new role and specify the following permissions.

   - Under **Global**, select **Cancel task**.
   - Under **Tasks**, select **Create task** and **Update task**.
   - Under **vApp**, select **Export** and **View OVF Environment**.

   Click **OK**.

3. To grant the new role permission to vCenter users or groups who will be importing virtual machines to Amazon EC2, right-click your specific vCenter in the tree-view pane and select **Add permission**. The **Assign Permissions** dialog box opens.

4. In the **Users and Groups** box on the left, select the users or groups you want to add to the new role you created.
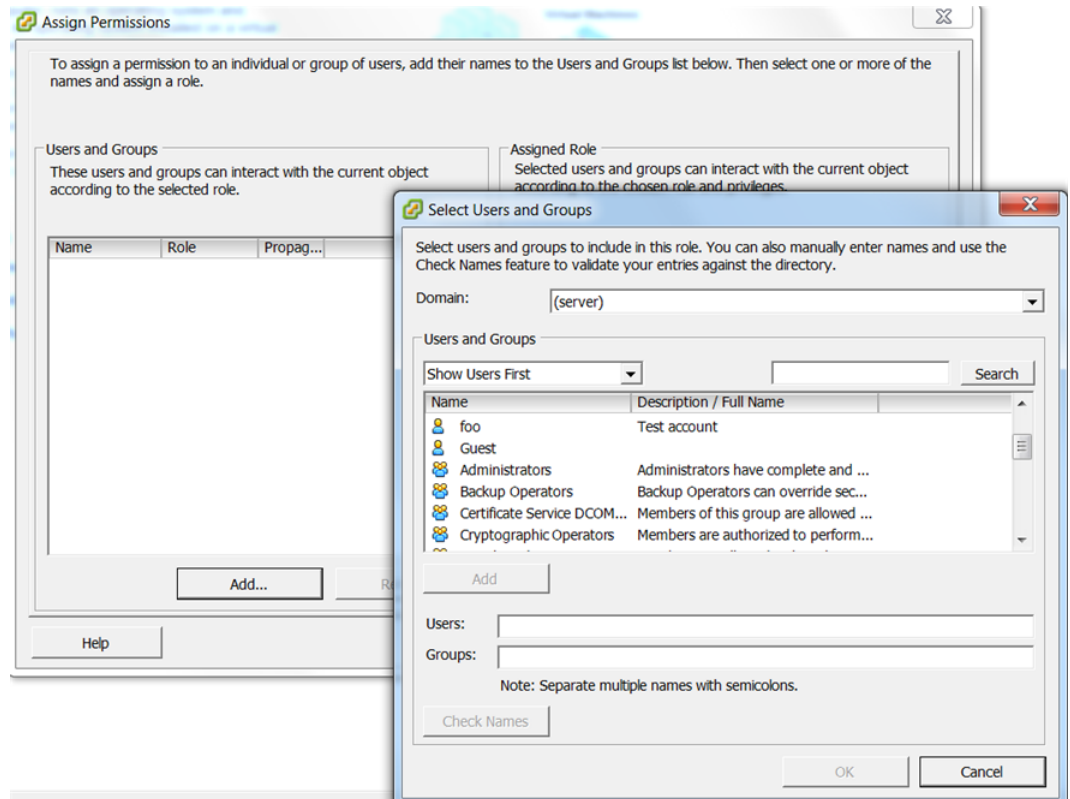
If you don't have users defined yet, click **Add**. The **Select Users and Groups** dialog box opens.

5. Select and add the users you want to add to the new role. When you have identified all the users for the role, click **OK**.

6. In the **Assigned Role** box on the right of the **Assign Permissions** dialog box, select the role that you previously created.

7. Clear the check box for **Propagate to Child Objects** and click **OK**.

   Repeat the same process to assign the role of virtual machine power user to all users and groups that you want to allow to import VMs to Amazon EC2. You can do this at the VM object level or at a higher level in the hierarchy. If you assign the role at a higher level, you must select the check box for **Propagate to Child Objects**.
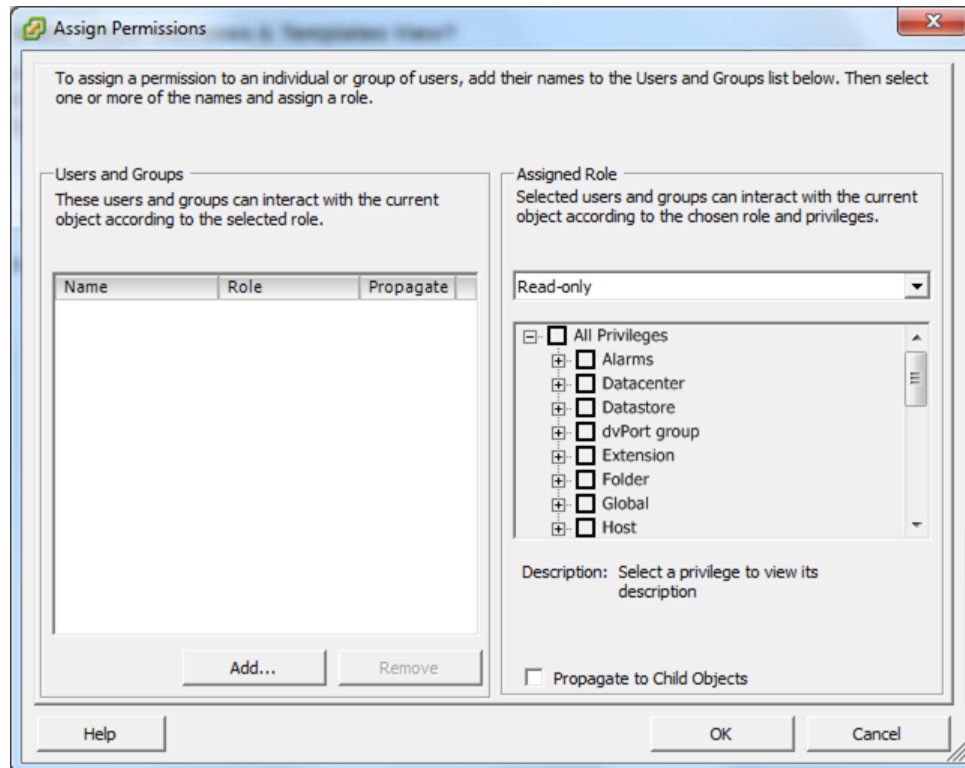
Using the Connector for VMware vCenter

This section shows you how to use the Connector to import a virtual machine to Amazon EC2 for the first time using an account with administrative rights.

Confirm that you have prepared the virtual machine according to the guidelines in Preparing Your Virtual Machine (p. 226).

Important

If you don't enable RDP and disable the Windows-based firewall, your VM will import successfully to Amazon EC2, but you will not be able to log in.

These requirements must be satisfied:

- The virtual machine must be turned off.
- The virtual machine must only use a single virtual hard drive (multiple partitions are OK).
- The virtual hard drive cannot be larger than one terabyte (1TB).
- The Connector virtual appliance must have sufficient free hard drive space to temporarily store the compressed VMDK image while it is being imported to Amazon EC2.

To use the Connector to import a VM for the first time using an account with administrative rights

1. Log in to VMware vCenter using the VMware vSphere Client. If you had a session open while you were installing the Connector, notice that the **Import to EC2** tab becomes visible.

Note

> The first time you log in, you will see an SSL certificate warning. This warning indicates that the SSL certificate being used by the Connector cannot be verified by an external source. This is expected behavior. Your session will continue to use SSL for encryption. Check the **Install this certificate and do not display a security warning** option at the bottom of the screen and click **Ignore**.

You are now logged in to the vCenter Server.

2. On the left pane, navigate the tree view to the virtual machine you want to import. Select it.

3. On the right pane, select the **Import to EC2** tab.

Note

> You might see another security warning about SSL certificates. Click **Yes** to continue.

The Connector initializes.

4. In the **Enter AWS Credentials** dialog box, provide your **Access Key ID** and **Secret Access Key**. For information on how to get your AWS credentials, see How to Get Your Access Key ID and Secret Access Key (p. 352).

The main Connector page appears when your AWS credentials are verified.

5. Back in vCenter, select the virtual machine you want to import and go to the **Import to EC2** tab.
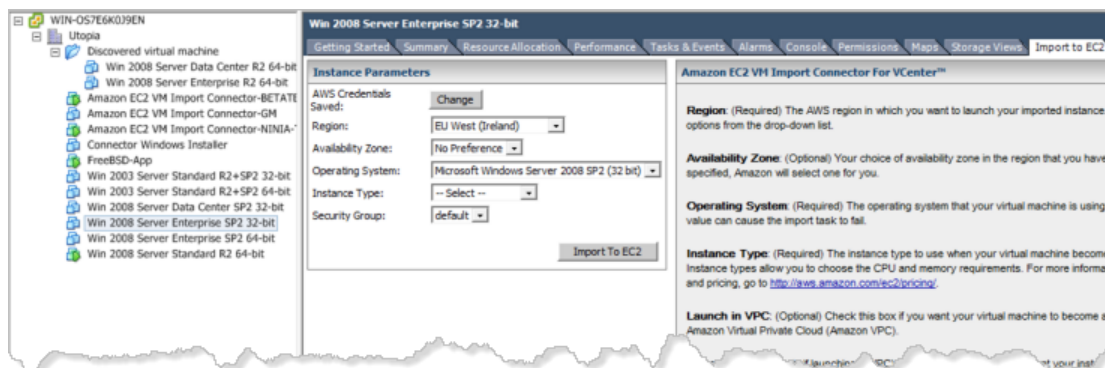
6. In the **Instance Parameters** dialog box, specify the values for the following options, then click the **Import to EC2** button.

- **Region**—(Required)The AWS Region in which you want to launch your imported instance. Select one of the options from the drop-down list.
- **Availability Zone**—(Optional) Your choice of Availability Zone in the Region that you have selected. If not specified, Amazon will select one for you.
- **Operating System**—(Required) The operating system that your virtual machine is using. Selecting an incorrect value can cause the import task to fail.

  For virtual machines using Windows Server 2008 R2, always select **Microsoft Windows Server 2008 (64-bit)**. If the virtual machine you're importing runs on Windows Server 2008 SP2 (also called Windows Server 2008 R1), determine first whether the operating system is a 32-bit or 64-bit **System Type** then select the **Operating System** accordingly. For more information about 32-bit and 64-bit Windows, go to 32-bit and 64-bit Windows: frequently asked questions. For more information about how to determine System Type for Windows Server 2003, go to Microsoft Support.

- **Instance Type—**(Required) The instance type to use when your virtual machine becomes an instance in EC2. Instance types allow you to choose the CPU and memory requirements. For more information on instance types and pricing, go to Amazon EC2 Pricing.
- **Launch in VPC—**(Optional) Check this box if you want your virtual machine to become an instance within Amazon Virtual Private Cloud (Amazon VPC). For information, go to Amazon Virtual Private Cloud (Amazon VPC).
- **Subnet—**(Required only if launching in VPC) Select the subnet that you want your instance placed within in a VPC.
- **Private IP address—**(Optional, applies only if launching in VPC) Specify the private IP address of your instance within VPC.
- **Security Group—**(Required) Select the security group to use with your instance. Defaults to the default security group.

The values you specified are listed in the **Confirm Import Options** box. Check the information and click **Import**.

7. Monitor the progress of the import task in the **Tasks & Events** or **Recent Tasks** tab of the VMware vSphere Client.
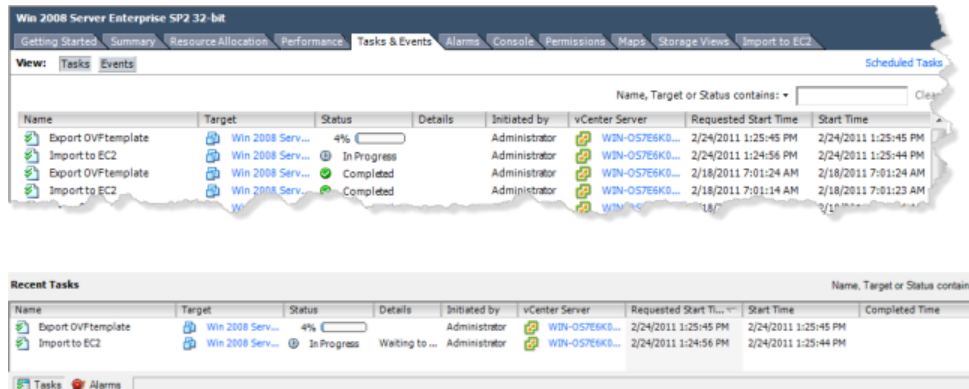
- **Export OVF template—**Creates a stream-optimized VMDK image. This process consolidates your virtual machine to a single image. In addition, stream-optimized VMDKs are compressed and are well-suited for transfer over a WAN connection. The stream-optimized VMDK will be temporarily stored on your Connector virtual appliance.
- **Import to EC2—**Transfers the stream-optimized VMDK that was created in the first task to Amazon EC2, and converts your virtual machine to an Amazon EC2 instance.

The **Import to EC2** task can take up to a few hours to complete. In addition, you might notice that the task progress will pause for up to 10 minutes at times. This is expected behavior.

Important

Keep your session in VMware vCenter open until all tasks complete. If you quit the vSphere Client or log off of your vCenter, the import task will not complete successfully, and the progress indicator will not be updated. If this occurs, you can use the command line tools to check the status of your import task. For more information, see Checking on the Status of Your Import in Using the Command Line Tools to Import Your Virtual Machine to Amazon EC2 (p. 240). When you have verified that the task is complete, you can safely cancel the import task by right-clicking the task in the vSphere Client and selecting **Cancel**.

Although you can see your instance in the AWS Management Console when the import process begins, do not launch your instance until the import process completes. For information about launching instances, see Launching an Instance from an AMI (p. 217).

Uninstalling the Connector for VMware vCenter

If you no longer want to use the Connector for VMware vCenter and you want to uninstall the virtual appliance, you will follow a two-part process:

• Deregister the Connector from the vCenter Server.
• Shut down the virtual appliance.

To uninstall the Connector from the VMware vCenter

1. Open a web connection to the Connector's IP address and log in.
2. Click the **vCenter** tab, then click **Deregister**.
3. In the **vCenter Connector Deregistration** page, enter the vCenter IP information and user name and password, then click **Deregister Connector with vCenter**.
4. When the Connector is no longer registered with the vCenter Server, shut down the virtual appliance and remove it from vCenter.

Using the Command Line Tools to Import Your Virtual Machine to Amazon EC2

Topics

In this section, you'll learn how to use the Amazon EC2 command line tools to import your Citrix, Microsoft Hyper-V, or VMware virtual machine to Amazon EC2.

Importing VMs into Amazon EC2 is a two-step process. First, you export your virtual machine from the virtualization environment. Next, you create an import task and upload your virtual machine into Amazon EC2. The following diagram illustrates this process.
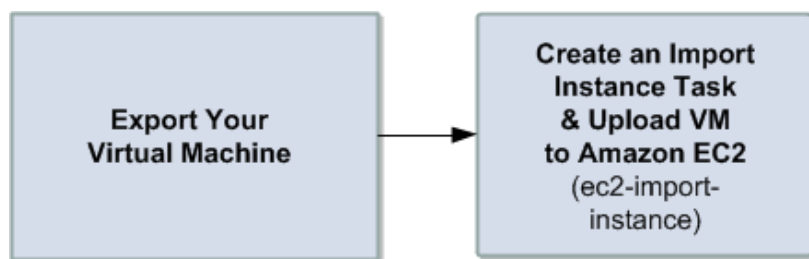
Use the following commands when you perform import tasks using the Amazon EC2 command line tools:

Command	Description
ec2-import-instance	Creates a new import instance task using metadata from the specified disk image and imports the instance to Amazon EC2.

Command	Description
`ec2-import-volume`	Creates a new import volume task using metadata from the specified disk image and imports the volume to Amazon EC2.
`ec2-resume-import`	Resumes the upload of a disk image associated with an import instance or import volume task ID.
`ec2-describe-conversion-tasks`	Lists and describes your conversion tasks.
`ec2-cancel-conversion-task`	Cancels the active conversion task. The task can be the import of an instance or volume.
`ec2-delete-disk-image`	Deletes a partially or fully uploaded disk image for conversion from Amazon S3.

For information about these commands and other EC2 commands, go to the Amazon Elastic Compute Cloud Command Line Reference.

Note

You can use a graphical user interface provided through the Amazon EC2 VM Import Connector (Connector) to import your VMware virtual machines to Amazon EC2. For more information, see Using the Amazon EC2 VM Import Connector to Import Your Virtual Machine to Amazon EC2 (p. 226). You cannot use the Connector to import Citrix or Microsoft Hyper-V virtual machines.

Exporting Your Virtual Machines from Their Virtual Environment

The process of exporting virtual machines depends on the source virtualization environment. In this section, we will show you the basic steps to export virtual machines from Citrix, Microsoft Hyper-V, and VMware virtualization products. For in-depth information, consult the documentation for these products.

- Exporting from Citrix (p. 241)
- Exporting from Microsoft Hyper-V (p. 244)
- Exporting from VMware (p. 246)

Before starting the export process, prepare the Windows Server environment of the virtual machine you are exporting so that:

- Remote desktop is enabled.
- Windows firewall allows public RDP traffic.
- Autologon is disabled.
- There are no pending Microsoft updates and the computer is not set to install software when it reboots.

Exporting from Citrix

Before you export your virtual machines from Citrix XenCenter, you must perform the following tasks:

- Remove the **Citrix Tools for Virtual Machines** from the VM.
- Use the Citrix XenCenter console to remove the tools.

If you have multiple virtual disks that you want to export from Citrix Xen, we recommend that you export the disks one at a time. The export of multiple virtual disks at the same time results in a list of randomly named VHD files.

To remove Citrix tools for virtual machines

1. In **XenCenter**, select the virtual machine you want to export, and click the **Console** tab, which shows the Windows desktop of the virtual machine.
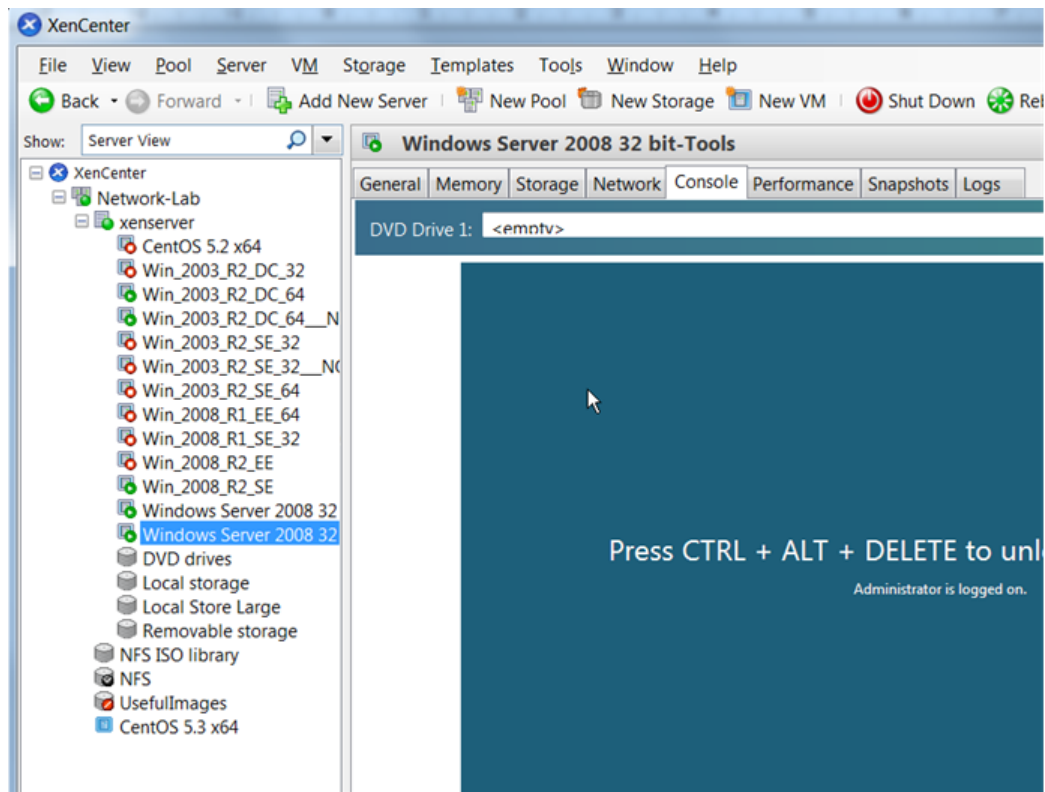
2. Using the **Console**, access the Control Panel of the virtual machine's Windows operating system and uninstall the **Citrix Tools for Virtual Machines**.
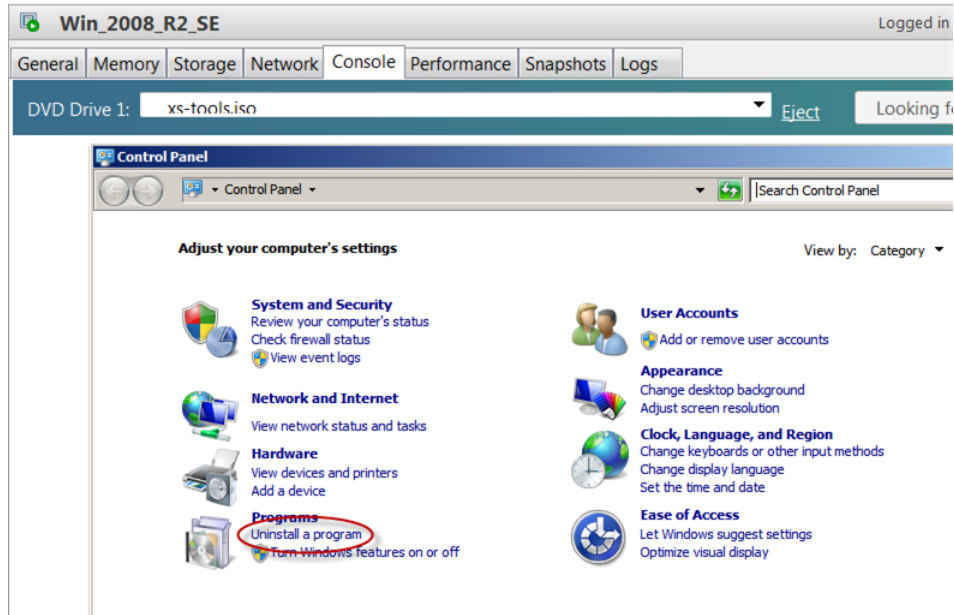
3. After the tools are removed, reboot the virtual machine when prompted, log in again and then shut down using Windows.
You can now proceed to export the VM.

To export an image from Citrix

1. In the Citrix XenCenter, select the virtual machine you want to export, and then go to the **Tools** menu, click **Virtual Appliance Tools**, and then **Export Appliance**.

   Note

   Do not export the image by right-clicking a stopped instance and selecting "Export to File."
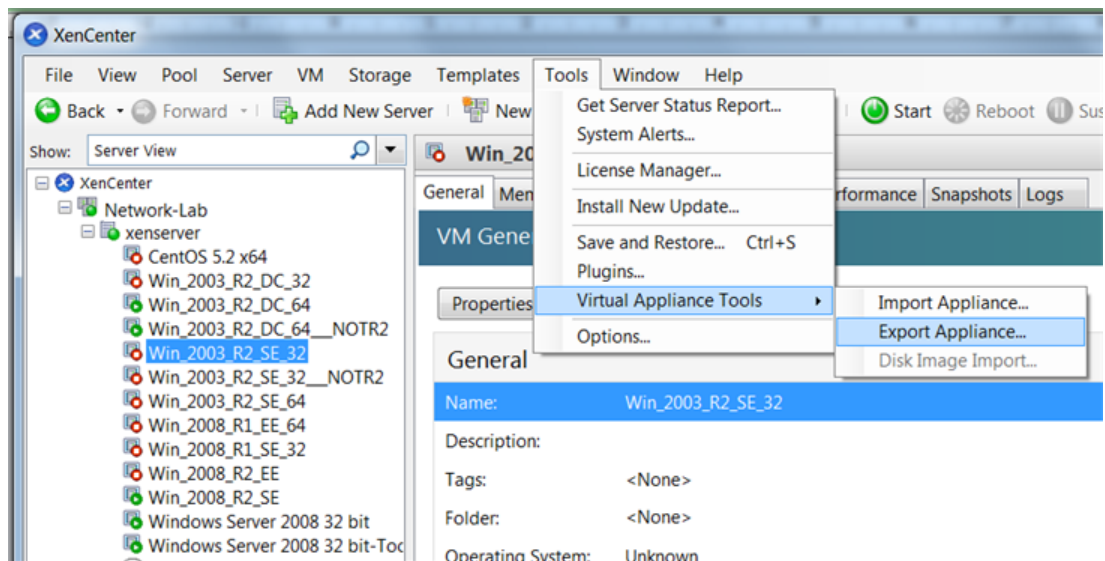
2. When the **XenServer OVF Appliance Export Wizard** starts, specify the destination of the VM files, click **Next**, accept the defaults, and proceed through the screens until you click **Finish**.
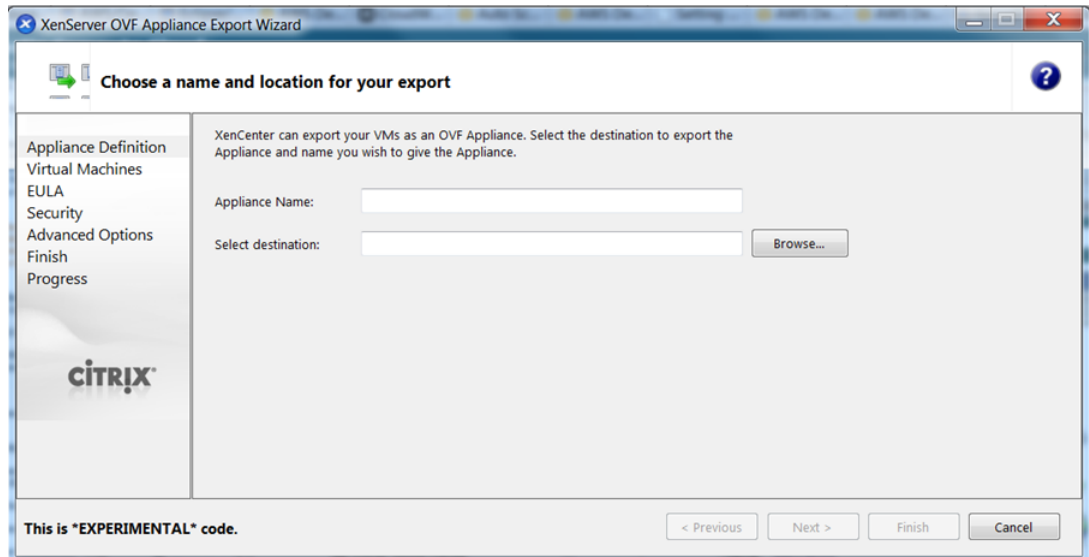
3. When the export completes, you can proceed and import the VM files to Amazon EC2 by following the steps in Importing Your Virtual Machine into Amazon EC2 (p. 247) and specifying VHD as the file format.

Exporting from Microsoft Hyper-V

To export Hyper-V virtual disks from Microsoft, you use the Hyper-V Manager.

To export a Hyper-V image from Microsoft

1. In the **Hyper-V Manager**, shut down the virtual machine you want to export.
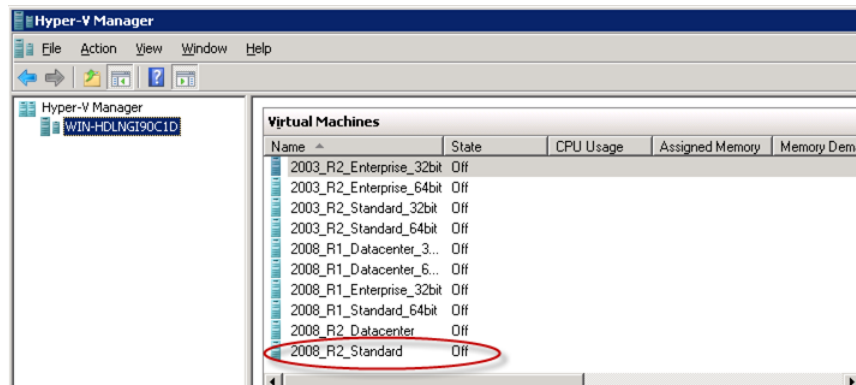
2. In the **Actions** pane for the virtual machine, select **Export**.
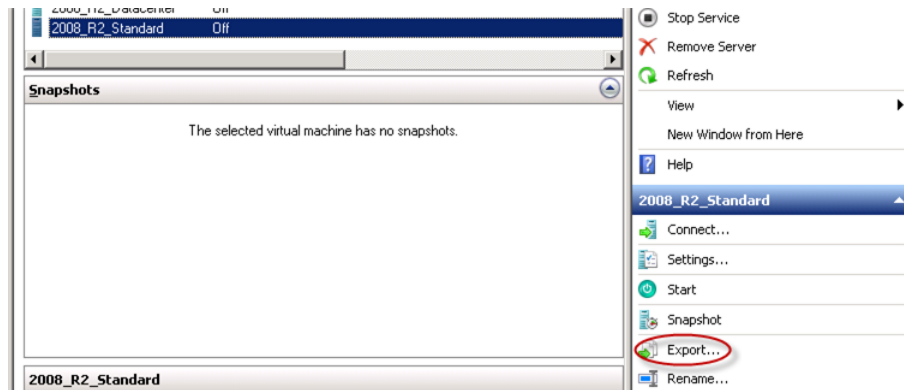
3. In the **Export Virtual Machine** dialog box, for **Location**, click **Browse**, navigate to a destination location that has plenty of space, and click **Export**.
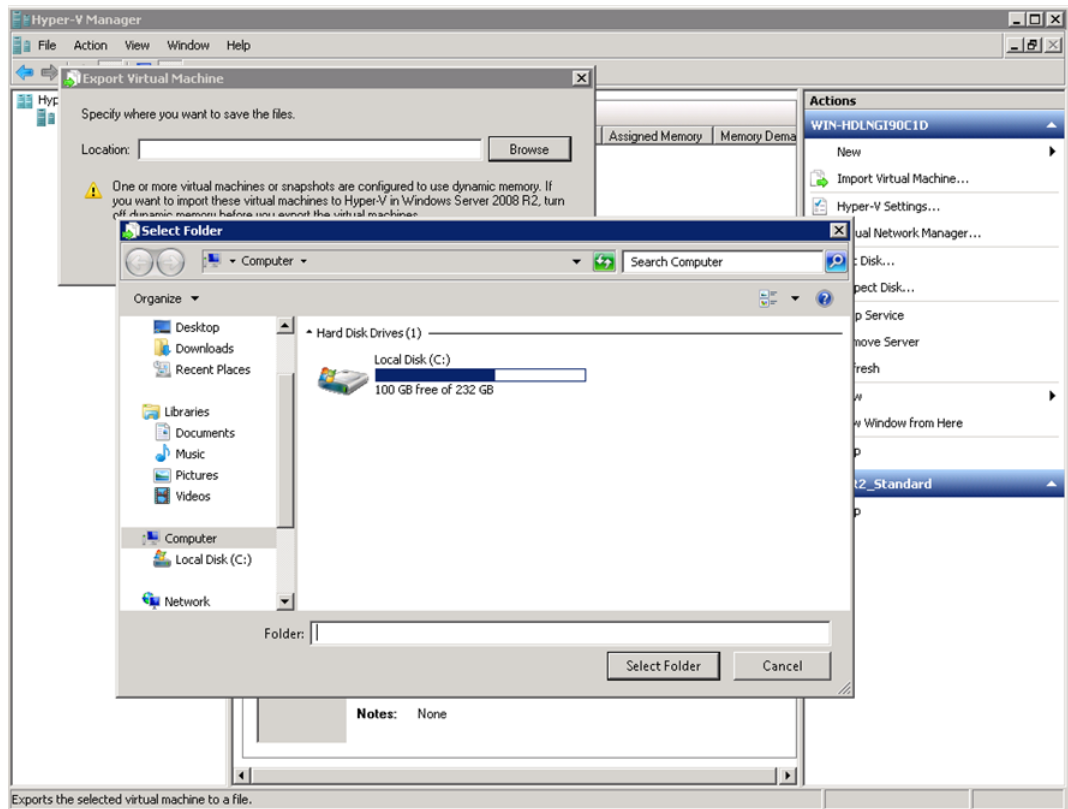
4. Track the export progress through the **Status** of your VM in the Hyper-V Manager. Wait for the export to complete.

    When the export completes, you can proceed and import the VM files to Amazon EC2 by following the steps in Importing Your Virtual Machine into Amazon EC2 (p. 247) and specifying VHD as the file format. The VHD file will be located in the folder you specified in the **Export Virtual Machine** dialog box.

Exporting from VMware

Before you can import a VMware vSphere VM or volume into Amazon EC2, you must export the VMDK disk image file. The following procedure shows you how to use the VMware vSphere Client to export a VM (and VMDK file). For more detailed information, consult your VMware documentation.

To export a disk image from VMware

1. In the VMware vSphere Client, select the virtual machine to export.
2. Shut down the machine.
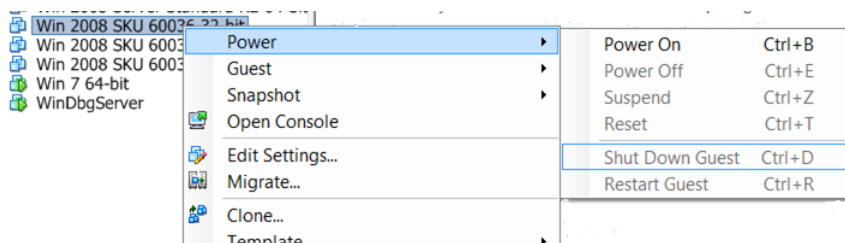
3. From the **File** menu, select **Export**, and then **Export OVF Template**.
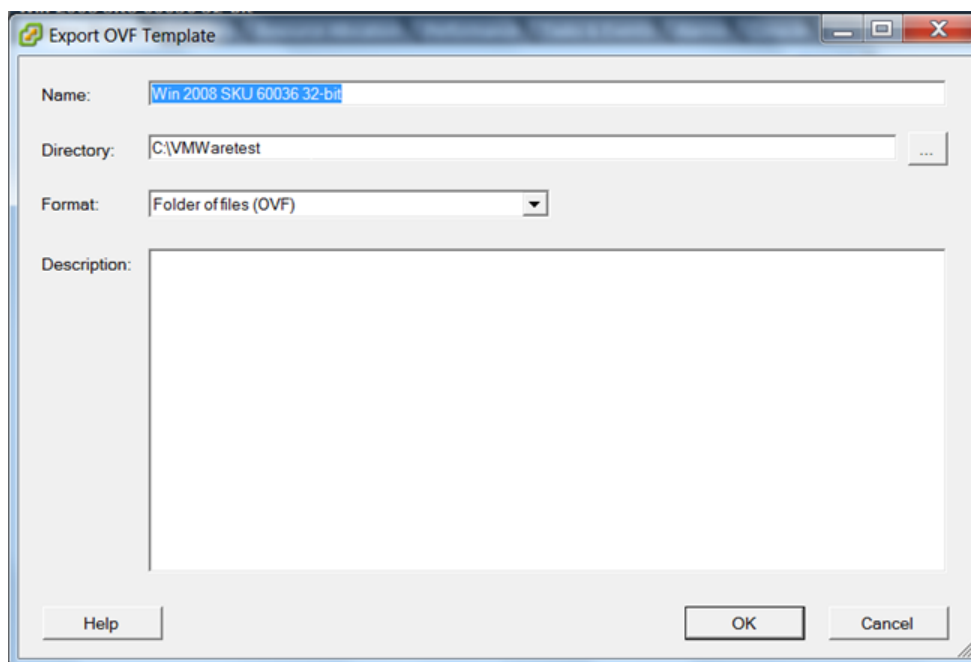
4. In the **Export OVF Template** dialog box, enter a name for the disk image file and a directory to save it in.
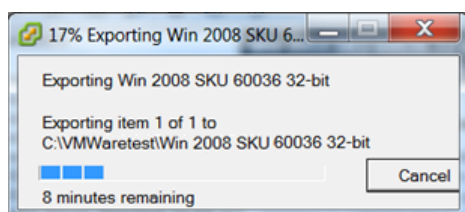
You will see a box displaying the progress of the export. Because of the size and number of files associated with the VM, and the network connection, the export might take some time. If the connection times out, the export process will fail. If this happens, restart the export. When the export completes, vSphere saves the disk image file in the directory you specified. Use the name of the VMDK file as an argument in `ec2-import-instance` to import a virtual machine into Amazon EC2, or in `ec2-import-volume` to import a volume into Amazon Elastic Block Store (Amazon EBS).

Importing Your Virtual Machine into Amazon EC2

After exporting your virtual machine from the third-party virtual environment, you can import it into Amazon EC2. The import process is the same regardless of the origin of the virtual machine.

Here are some important things to know about your VM instance, as well as some security and storage recommendations:

- Amazon EC2 automatically assigns a DHCP IP address to your instance. The DNS name and IP address are available via the `ec2-describe-instances` command when the instance starts running.
- Your instance has only one Ethernet network interface.
- If you didn't specify an Amazon Virtual Private Cloud (Amazon VPC) subnet to use when you created the conversion task, your instance has a public IP address. We recommend you use a restrictive security group to control access to your instance.
- We recommend that your instance contain strong passwords for all user accounts.

To import a virtual machine

You can import a virtual machine into Amazon EC2. If the import of the virtual machine is interrupted, you can use the ec-2-resume-import command to resume the import from where it stopped. For more information, see Resuming an Upload (p. 251).

- Use `ec2-import-instance` to create a new import instance task.
  The syntax of the command is:

```
ec2-import-instance DISK_IMAGE_FILENAME -t INSTANCETYPE -f FORMAT -a ARCHI
TECTURE-SYSTEM -b S3_BUCKET_NAME -o OWNER -w SECRETKEY
```

The following command creates an import instance task that imports a Windows Server 2008 SP2 (32-bit) VM into the us-east-1 Region.

Note

The example uses the VMDK format. You can also use VHD or RAW.

```
ec2-import-instance ./WinSvr8-2-32-disk1.vmdk  -f VMDK -t m1.small -a i386
 -b myawsbucket -o AKIAIOSFODNN7EXAMPLE -w wJalrXUtnFEMI/K7MDENG/bPxRfi
CYEXAMPLEKEY
```

This request uses the VMDK file, `WinSvr8-2-32-disk1.vmdk`, to create the import task. The output is similar to the following example.

```
Requesting volume size: 25 GB
Disk image format: Stream-optimized VMDK
Converted volume size: 26843545600 bytes (25.00 GiB)
Requested EBS volume size: 26843545600 bytes (25.00 GiB)
```

```
TaskType          IMPORTINSTANCE  TaskId  import-i-fhbx6hua      ExpirationTime
   2011-09-09T15:03:38+00:00        Status  active  StatusMessage    Pending
InstanceID       i-6ced060c
DISKIMAGE        DiskImageFormat VMDK    DiskImageSize    5070303744
VolumeSize       25      AvailabilityZone        us-east-1c      Approximate
BytesConverted        0       Status   active StatusMessage    Pending
Creating new manifest at testImport/9cba4345-b73e-4469-8106-
2756a9f5a077/Win_2008_R1_EE_64.vmdkmanifest.xml
Uploading the manifest file
Uploading 5070303744 bytes across 484 parts
0% |-------------------------------------------------| 100%
    |=================================================|
Done
```

Checking on the Status of Your Import

The `ec2-describe-conversion-tasks` command returns the status of an import. Status values
include:

- **active—**Your instance or volume is still importing.
- **cancelling—**Your instance or volume is still being canceled.
- **cancelled—**Your instance or volume is canceled.
- **completed—**Your instance or volume is ready to use.
  The imported instance is in the stopped state. You use `ec2-start-instance` to start it. For more
  information, go to ec2-start-instances in the *Amazon Elastic Compute Cloud Command Line Reference*.

To check the status of your import

- Use `ec2-describe-conversion-task` to return the status of the task. The syntax of the command
  is:

```
ec2-describe-conversion-tasks TASKID
```

  The following example enables you to see the status of your import instance task.

```
ec2-describe-conversion-tasks import-i-ffvko9js
```

  The following response shows that the IMPORTINSTANCE status is `active`, and 73747456 bytes
  out of 893968896 have been converted.

```
TaskType          IMPORTINSTANCE  TaskId  import-i-ffvko9js      ExpirationTime
   2011-06-07T13:30:50+00:00        Status   active  StatusMessage    Pending
InstanceID       i-17912579
DISKIMAGE        DiskImageFormat VMDK    DiskImageSize   893968896 VolumeSize
     12      AvailabilityZone        us-east-1a      ApproximateBytesCon
verted       73747456      Status   active StatusMessage    Pending
```

  The following response shows that the IMPORTINSTANCE status is `active`, and at 7% progress and
  that the DISKIMAGE is completed.

```
TaskType        IMPORTINSTANCE  TaskId import-i-ffvko9js      ExpirationTime
  2011-06-07T13:30:50+00:00       Status  active  StatusMessage   Progress:
 7%     InstanceID      i-17912579
DISKIMAGE       DiskImageFormat VMDK    DiskImageSize    893968896 VolumeId
      vol-9b59daf0    VolumeSize      12      AvailabilityZone        us-
east-1a     ApproximateBytesConverted       893968896 Status  completed
```

The following response shows that the `IMPORTINSTANCE` status is `completed`.

```
TaskType        IMPORTINSTANCE  TaskId import-i-ffvko9js      ExpirationTime
  2011-06-07T13:30:50+00:00       Status  completed   InstanceID    i-
17912579
DISKIMAGE       DiskImageFormat VMDK    DiskImageSize    893968896 VolumeId
      vol-9b59daf0    VolumeSize      12      AvailabilityZone        us-
east-1a     ApproximateBytesConverted       893968896 Status  completed
```

Note

The `IMPORTINSTANCE` status is what you use to determine the final status. The `DISKIMAGE` status will be `completed` for a period of time before the `IMPORTINSTANCE` status is `completed`.

You can now use commands such as `ec2-stop-instance`, `ec2-start-instance`, `ec2-reboot-instance`, and `ec2-terminate-instance` to manage your instance.

Note

By default, when you terminate an instance, Amazon EC2 does not delete the associated Amazon EBS volume. You can optionally use the `ec2-modify-instance-attribute` command to change this behavior.

Importing Your Volumes into Amazon EBS

This section describes how to import your data storage into Amazon EBS, and then attach it to one of your existing Amazon EC2 instances. Amazon EC2 supports importing RAW and VMDK disk formats.

Important

We recommend utilizing Amazon EC2 security groups to limit network access to your imported instance. Configure a security group to allow only trusted Amazon EC2 instances and remote hosts to connect to RDP and other service ports. For more information about security groups, see Security Groups (p. 416).

After you have exported your virtual machine from the virtualization environment, importing the volume to Amazon EBS is a single-step process. You create an import task and upload the volume in one step, as illustrated in the following diagram.
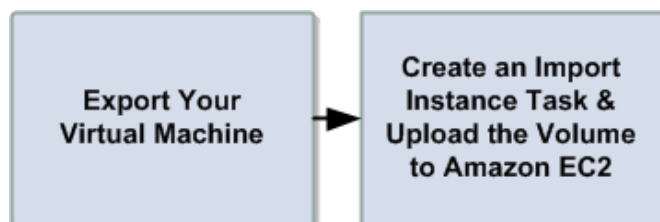
To import a volume into Amazon EBS

1. Use `ec2-import-volume` to create a task that allows you to upload your volume into Amazon EBS. The syntax of the command is:

    ```
    $ ec2-import-volume DISK_IMAGE_FILENAME -f FORMAT -s SIZE-IN-GB -z AVAILAB
    ILITY_ZONE -b S3_BUCKET_NAME -o OWNER -w SECRETKEY
    ```

    The following example creates an import volume task for importing a volume to the us-east-1a Region.

    > **Note**
    >
    > The example uses the VMDK format. You can also use VHD or RAW.

    ```
    Requesting volume size: 25 GB
    Disk image format: Stream-optimized VMDK
    Converted volume size: 26843545600 bytes (25.00 GiB)
    Requested EBS volume size: 26843545600 bytes (25.00 GiB)
    TaskType        IMPORTVOLUME    TaskId import-vol-ffut5xv4     ExpirationTime
      2011-09-09T15:22:30+00:00        Status  active  StatusMessage   Pending
    DISKIMAGE       DiskImageFormat VMDK    DiskImageSize   5070303744
    VolumeSize      25      AvailabilityZone        us-east-1d      Approximate
    BytesConverted      0
    Creating new manifest at tesdfgting/0fd8fcf5-04d8-44ae-981f-
    3c9f56d04520/Win_2008_R1_EE_64.vmdkmanifest.xml
    Uploading the manifest file
    Uploading 5070303744 bytes across 484 parts
    0% |--------------------------------------------------| 100%
       |==================================================|
    Done
    ```

    Amazon EC2 returns a task ID that you use in the next step. In this example, the ID is `import-vol-ffut5xv4`.

2. Use `ec2-describe-conversion-tasks` to confirm that your volume imported successfully.

    ```
    $ ec2-describe-conversion-tasks import-vol-ffut5xv4
    TaskType        IMPORTVOLUME    TaskId import-vol-ffut5xv4     ExpirationTime
      2011-09-09T15:22:30+00:00        Status  completed
    DISKIMAGE       DiskImageFormat VMDK    DiskImageSize   5070303744
    VolumeId        vol-365a385c    VolumeSize      25      AvailabilityZone
          us-east-1d      ApproximateBytesConverted       5070303744
    ```

    The status in this example is `completed`, which means the import succeeded.

3. Use `ec2-attach-volume` to attach the Amazon EBS volume to one of your existing Amazon EC2 instances. The following example attaches the volume, `vol-2540994c`, to the `i-a149ec4a` instance on the device, `/dev/sde`.

    ```
    $ ec2-attach-volume vol-2540994c -i i-a149ec4a -d /dev/sde
    ATTACHMENT vol-2540994c i-a149ec4a /dev/sde attaching 2010-03-
    23T15:43:46+00:00
    ```

Resuming an Upload

Connectivity problems can interrupt an upload. When you resume an upload, Amazon EC2 automatically starts the upload from where it stopped. The following procedure steps you through determining how much of an upload succeeded and how to resume it.

To resume an upload

* Use the task ID with `ec2-resume-import` to continue the upload. The command uses the HTTP `HEAD` action to determine where to resume.

```
ec2-resume-import DISK_IMAGE_FILENAME -t TASK_ID -o OWNER -w SECRETKEY
```

The following example resumes an import instance task.

```
Disk image size: 5070303744 bytes (4.72 GiB)
Disk image format: Stream-optimized VMDK
Converted volume size: 26843545600 bytes (25.00 GiB)
Requested EBS volume size: 26843545600 bytes (25.00 GiB)
Uploading 5070303744 bytes across 484 parts
0% |--------------------------------------------------| 100%
   |==================================================|
Done
Average speed was 10.316 MBps
The disk image for import-i-ffni8aei has been uploaded to Amazon S3
where it is being converted into an EC2 instance.  You may monitor the
progress of this task by running ec2-describe-conversion-tasks.  When
the task is completed, you may use ec2-delete-disk-image to remove the
image from S3.
```

Canceling an Upload

Use `ec2-cancel-conversion-task` to cancel an active conversion task. The task can be the upload of an instance or a volume. The command removes all artifacts of the conversion, including uploaded volumes or instances.

If the conversion is complete or still transferring the final disk image, the command fails and returns an exception similar to the following:

```
Client.CancelConversionTask Error: Failed to cancel conversion task import-i-
fh95npoc
```

To cancel a conversion task

* Use the task ID of the upload you want to delete with `ec2-cancel-conversion-task`. The following example cancels the upload associated with the task ID `import-i-fh95npoc`.

```
PROMPT> ec2-cancel-conversion-task import-i-fh95npoc
```

The output for a successful cancellation is similar to the following:

```
CONVERSION-TASK import-i-fh95npoc
```

You can use the `ec2-describe-conversion-tasks` command to check the status of the cancellation. For example:

```
$ ./ec2-describe-conversion-tasks import-i-fh95npoc
TaskType         IMPORTINSTANCE  TaskId  import-i-fh95npoc        ExpirationTime
  2010-12-20T18:36:39+00:00     Status  cancelled   InstanceID      i-825063ef
DISKIMAGE        DiskImageFormat VMDK    DiskImageSize   2671981568
VolumeSize     40      AvailabilityZone      us-east-1c  ApproximateBytesCon
verted       0       Status  cancelled
```

In the above example, the status is *cancelled*. If it were still in process, the status would be *cancelling*.

Cleaning up After an Upload

You can use `ec2-delete-disk-image` to remove the image file after it is uploaded. If you do not delete it, you will be charged for its storage in Amazon S3.

To delete a disk image

* Use the task ID of the disk image you want to delete with `ec2-delete-disk-image`.

  The following example deletes the disk image associated with the task ID, `import-i-fh95npoc`.

  ```
  PROMPT> ec2-delete-disk-image import-i-fh95npoc
  ```

  The output for a successful cancellation is similar to the following:

  ```
  DELETE-TASK import-i-fh95npoc
  ```

Exporting EC2 Instances

Topics

If you have previously imported an instance into Amazon Elastic Compute Cloud (Amazon EC2), you can use the command line tools to export the instance to Citrix Xen, Microsoft Hyper-V, or VMware vSphere. Exporting an instance can be useful when you want to deploy a copy of your EC2 instance in your on-site virtualization environment.

Before You Get Started

Before you begin the process of exporting an instance, you need to be aware of the operating systems and image formats we support, and understand the limitations on exporting instances and volumes. You will also need to download and install the EC2 command line tools and sign up for your private key and X.509 certificate before you use the command line interface (CLI) or the API to export your instance. For more information, see Setting Up the Amazon EC2 Command Line Tools (p. 522).

Operating Systems

The following operating systems can be exported from Amazon EC2:

- Windows Server 2003 R2 (Standard, Enterprise, and Datacenter)
- Windows Server 2008 R1 and R2 (Standard, Enterprise, and Datacenter)

Image Formats

We support the following image formats for exporting both volumes and instances from Amazon Web Services (AWS):

- Stream-optimized ESX Virtual Machine Disk (VMDK) image format, which is compatible with VMware ESX and VMware vSphere versions 4 and 5 virtualization products.
- Open Virtual Appliance (OVA) image format, which is compatible with VMware vSphere versions 4 and 5.
- Virtual Hard Disk (VHD) image format, which is compatible with Citrix Xen and Microsoft Hyper-V virtualization products.

Known Limitations

The exporting of instances and volumes is subject to the following limitations:

- You cannot export Amazon Elastic Block Store (Amazon EBS) data volumes.
- You cannot export an instance that has more than one virtual disk.
- You cannot export an instance that has more than one network interface.

Export an Instance

You can use the Amazon EC2 command line interface (CLI) to export an instance. The ec2-create-instance-export-task command gathers all of the information necessary (e.g., instance ID; name of the S3 bucket that will hold the exported image; name of the exported image; VMDK, OVA, or VHD format) to properly export the instance to the selected virtualzation format. The exported file is saved in the Amazon Simple Storage Service (Amazon S3) bucket that you designate.

> **Note**
>
> When you export an instance, you are charged the standard Amazon S3 rates for the bucket where the exported VM is stored. In addition, a small charge reflecting temporary use of an EBS snapshot might appear on your bill. For more information about Amazon S3 pricing, see Amazon Simple Storage Service (S3) Pricing.

To export an instance

1. Create an Amazon S3 bucket where exported instances will be stored. The S3 bucket must grant **Upload/Delete** and **View Permissions** access to the **vm-import-export@amazon.com** account. For more information, see Creating a Bucket and Editing Bucket Permissions in the Amazon Simple Storage Service User Guide.
2. At a command prompt, type the following command: `ec2-create-instance-export-task`
   `INSTANCE_ID -e TARGET_ENVIRONMENT -f DISK_IMAGE_FORMAT -c CONTAINER_FORMAT`
   `-b S3_BUCKET`

   Where:

   `INSTANCE_ID` is the ID of the instance you want to export.

   `TARGET_ENVIRONMENT` is VMware, Citrix, or Microsoft.

   `DISK_IMAGE_FORMAT` is VMDK for VMware or VHD for Microsoft Hyper-V and Citrix Xen.

`CONTAINER_FORMAT` may be optionally set to OVA when exporting to VMware.

`S3_BUCKET` is the name of the Amazon S3 bucket to which you want to export the instance.

3. To monitor the export of your instance, at the command prompt, type the following command:
`ec2-describe-export-tasks TASK_ID`

Where:

`TASK_ID` is the ID of the export task.

Cancel or Stop the Export of an Instance

You can use the Amazon EC2 command line interface (CLI) to cancel or stop the export of an instance up to the point of completion. The ec2-cancel-export-task command removes all artifacts of the export, including any partially created Amazon S3 objects. If the export task is complete or is in the process of transferring the final disk image, the command fails and returns an error.

To cancel or stop the export of an instance

- At the command prompt, type: `ec2-cancel-export-task TASK_ID`

Where

`TASK_ID` is the ID of the export task you want to cancel.

Connecting to Amazon EC2 Instances

This section describes how to connect to instances that you launched and how to transfer files between your local computer and your Amazon EC2 instance. For information on launching instances, see Launching Amazon EC2 Instances (p. 217).

Prerequisites

- **Enable SSH/RDP traffic—**Open the instance's SSH or RDP port
  Before you try to connect, ensure that your Amazon EC2 instance accepts incoming traffic on the proper port. For Linux/UNIX instances, open port 22 for SSH access. For Windows instances, open port 3389 for RDP access. For more information, see Authorizing Network Access to Your Instances (p. 261).

- **SSH/RDP client—**Install an SSH or RDP client
  You can connect to Linux and UNIX computers using an SSH Java-based client with your web browser or a standalone SSH client. Most Linux and UNIX computers include an SSH client by default. You can check for an SSH client by typing **ssh** at the command line. If your computer doesn't recognize this command, the OpenSSH project provides a free implementation of the full suite of SSH tools. For more information, go to http://www.openssh.org. Likewise, most Windows computers include an RDP client. You can check for an RDP client by typing **mstsc** at the command line. If your computer doesn't recognize this command, go to the Microsoft Windows home page and search for the download for Remote Desktop Connection.

- **Instance ID—**Get the ID of your Amazon EC2 instance
  Retrieve the Instance ID of the Amazon EC2 instance you want to access. The Instance ID for all your instances are available in the AWS Management Console or through the CLI command ec2-describe-instances.

- **Private key—**Get the path to your private key
  You'll need the fully qualified path of the private key file associated with your instance. For more information on key pairs, see Getting an SSH Key Pair (p. 255).

To connect to an Amazon EC2 Linux/UNIX instance, use SSH. To connect to an Amazon EC2 Windows instance, use the Remote Desktop Protocol (RDP). The following topics demonstrate how to connect to your instance using the SSH and RDP protocols:

- Connecting to Linux/UNIX Instances Using SSH (p. 263)
- Connecting to Linux/UNIX Instances from Windows Using PuTTY (p. 269)
- Connecting to Windows Instances (p. 275)

Getting an SSH Key Pair

Topics
- How to Generate Your Own Key and Import It to AWS (p. 256)
- How to Have AWS Create the Key Pair for You (p. 257)

Public AMI instances have no password, and you need a public/private key pair to log in to them. The public key half of this pair is embedded in your instance, allowing you to use the private key to log in securely without a password. After you create your own AMIs, you can choose other mechanisms to securely log in to your new instances.

You can have multiple key pairs, and each key pair requires a name. Be sure to choose a name that is easy to remember.

You have two options for getting a key pair:

- Generate it yourself.

  You can use a third-party tool such as OpenSSH, and then import the public key to AWS using either the `ec2-import-keypair` command or the `ImportKeyPair` action.

- Have AWS generate it for you.

  You can use the AWS Management Console, the `ec2-create-keypair` command, or the `CreateKeyPair` action.

AWS doesn't store a copy of the private key for either option. Amazon EC2 only stores the public key, and associates it with a friendly name that you specify for the key pair.

> **Note**
>
> If you are using PuTTY in Windows, you must convert the private key to PuTTY's format. For more information on using PuTTy with Amazon EC2, see Connecting to Linux/UNIX Instances from Windows Using PuTTY (p. 269).

How to Generate Your Own Key and Import It to AWS

This section describes how to import a public key to AWS from a key pair you've created with a third-party tool.

You can easily create an RSA key pair on Windows or Linux using the `ssh-keygen` command line tool (provided with the standard OpenSSH installation). Java, Ruby, Python, and many other programming languages provide standard libraries for RSA key pair creation.

EC2 accepts the following formats:

- OpenSSH public key format (e.g., the format in ~/.ssh/authorized_keys)
- Base64 encoded DER format
- SSH public key file format as specified in RFC4716

EC2 does not accept DSA keys. Make sure your key generator is set up to create RSA keys.

Supported lengths: 1024, 2048, and 4096.

Command Line Tools

To import a public key

1. Generate the key pair with a third-party tool of your choice.
2. Use `ec2-import-keypair` to import the public key file to AWS. The following example names the key pair `gsg-keypair`. The response displays the MD5 public key fingerprint as specified in section 4 of RFC4716.

```
PROMPT>  ec2-import-keypair gsg-keypair --public-key-file C:\keys\mykey.ppk
KEYPAIR  gsg-keypair
00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
```

API

To import the public key

1. Generate the key pair with the third-party tool of your choice.
2. Use `ImportKeyPair` to import the public key file to AWS. The following Query example names the key pair `gsg-keypair`. You must base64 encode the public key material before sending it to AWS.

```
https://ec2.amazonaws.com/?Action=ImportKeyPair
&KeyName=gsg-keypair
&PublicKeyMaterial=LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tDQpNSUlDZHp
DQ0FlQ2dBd0lCQWdJR0FQalRy3pQ
TUEwR0NTcUdTSWIzRFFFQkJRVUFNRk14Q3pBSkJnTlZCQVlUQpbFZUVJNd0VRWURWUVFLRXdw
Qm1JXRjZiMjR1WTI5dE1Rd3dDZZ1lEV1FFRTEV3TkJWV1U14SVRBZkJnTlZCQVU1UDQpHRTUZYVXlCTWFX
MXBkR1ZrTFVGGemMzVnlaVzVpNCRFFUQWVGdzB3T1RBM016RXlNVFEzTXpwWYUZ3TXHhNREEzDQpN
ekV5TVRRM016VmVmFNRkl4Q3pBSkJnTlZCQVlUQVVMxWVE1STXdFUVlEV1FRS0V3cEJiV0Y2Yj1I0dVky
OXRNUmN3DQpGUVlEV1FRRTEV3TlNUJWMU10UkdWMlpXeHZjjR1Z5Y3pfVVk1CTURBMVVVFQXhhNTWJUSnNViV
RGhxZW00MWVHUjFNSUdmDQpNQTBHQ1NxR1NJYjNEUVVCQVFVQU0R05BRENaVFLQmdDRQ1dOazBo
QytrcExBRnRp2YkFQc3U1TDU5bFMwUnI0DQpprZEpaM0RFak1pL0IwV2ZSSszphS2hWWYt1WitHSnJt
NDdMUHZCaFVKWK9IeHVU0VXakFDNmlybDJzKzlSWXVjjDQpFZXg0TjI4ZlpCZGppORlAzdEgwZ2NuW2Nu
WjdIbXXZ4aFBBTRTdpZmViNmNGWUhRdHpHHRnRPQ0ZQTmdUSU92VDE5DQoyR3RZ1Zb1VyU3BDVGFC
UUlEQVFBQm8xY3dWVEFFPQmdOVkhROEJBZjhFQkFNQ0JoQXdEZ1lEVlIwbEVTRFRSC9CQQXd3DQpwDZllJ
S3dZQkJRVUhBd0l3REFZRFZSMFRBUUgvQkFJd0FEQVJCZ05WSFE0RUZnUVUuTUZZUzdyTDNX
TUdLDDQpqejMxVXZZ5TThnMHddEVUVlKS29aSWh2Y05BUUVGQlFBRGdZRUFnWjdDZ2llSJWHR1WFM1NHVq
bU5jjOTR0NWRNc3krrDQpCM0Z3WVVNdUg4WUI2eGGQvSUVWMTFLRVEyZ0hpcZUdMU21jUWg4c2JXTTdt
KzcrYm9UNmc2U2hLbU1jjblkzWkRTDQpWRVFZZ25qcEt1aEEZRd2pmaVpTUEc1UG5SVENhdkVqczlT
TUpDVGGxdpTdTTjMrR2J3cFU5Vzg3K21GM2tzMGRmDQpzTlIrbEl5SWcrU3ROOTg9DQotLS0tLUVO
RCBDRVJUSUZJQ0FURS0tLS0tEXAMPLE
&AuthParams
```

The response includes the MD5 public key fingerprint as specified in section 4 of [RFC4716](#).

```
<ImportKeyPairResponse xmlns="http://ec2.amazonaws.com/doc/2012-07-20/">
    <requestId>7a62c49f-347e-4fc4-9331-6e8eEXAMPLE</requestId>
    <keyName>gsg-keypair</keyName>
    <keyFingerprint>
      00:00:00: 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
    </keyFingerprint>
</ImportKeyPairResponse>
```

How to Have AWS Create the Key Pair for You

AWS Management Console

To generate a key pair

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Click **Key Pairs** in the **Navigation** pane.
   The console displays a list of key pairs associated with your account.
3. Click **Create Key Pair**.
   The **Key Pair** dialog box appears.
4. Enter a name for the new key pair in the **Key Pair Name** field and click **Create**.

You are prompted to download the key file.

5. Download the key file and keep it in a safe place. You need it to access any instances that you launch with this key pair.

Command Line Tools

Use the ec2-create-keypair command.

To generate a key pair

1. The following command names the resulting key pair `gsg-keypair`.

```
PROMPT>  ec2-create-keypair gsg-keypair
```

Amazon EC2 returns a private key, similar to this one.

```
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 ----
MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD
VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z
b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
rDHudUZg3qX4waLG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END RSA PRIVATE KEY-----
```

You must save the private key to a local file so that you can use it later.

2. Create a file named `id_rsa-gsg-keypair` and paste the entire key generated in step 1, including the following lines.

```
"---- BEGIN RSA PRIVATE KEY ----"
"-----END RSA PRIVATE KEY-----"
```

3. Confirm that the file contents looks similar to the following and save the file.

You can save the file in any directory, but if you do not put it in your current directory, you should specify the full path when using commands that require the key pair.

```
 ---- BEGIN RSA PRIVATE KEY ----
MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD
```

```
VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z
b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
rDHudUZg3qX4waLG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END RSA PRIVATE KEY-----
```

4. If you're using OpenSSH (or another SSH client), you should set the permissions of this file so it is only readable by you.

On Linux and UNIX, enter the information in the following example.

```
$   chmod 400 id_rsa-gsg-keypair ; ls -l id_rsa-gsg-keypair
```

You receive output similar to the following example.

```
-r--------  1 fred flintstones 1701 Jun 19 17:57 id_rsa-gsg-keypair
```

API

Use the CreateKeyPair action.

To generate a key pair

1. Construct the following Query request.

```
https://ec2.amazonaws.com/
?Action=CreateKeyPair
&KeyName=gsg-keypair
&AUTHPARAMS
```

The following is an example response.

```
<CreateKeyPairResponse xmlns="http://ec2.amazonaws.com/doc/2012-07-20/">
   <keyName>gsg-keypair</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 ----
MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD
VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z
b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
```

```
rDHudUZg3qX4waLG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END RSA PRIVATE KEY-----</keyMaterial>
</CreateKeyPairResponse>
```

You must save the private key to a local file so that you can use it later.

2. Create a file named `id_rsa-gsg-keypair` and paste the entire key generated in step 1, including the following lines.

    ```
    "---- BEGIN RSA PRIVATE KEY ----"
    "-----END RSA PRIVATE KEY-----"
    ```

3. Confirm that the file contents looks similar to the following and save the file.

    You can save the file in any directory, but if you do not put it in your current directory, you should specify the full path when using commands that require the key pair.

    ```
     ---- BEGIN RSA PRIVATE KEY ----
    MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
    VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
    b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
    BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
    MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD
    VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z
    b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
    YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
    21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
    rDHudUZg3qX4waLG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
    Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
    nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
    FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
    NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
    -----END RSA PRIVATE KEY-----
    ```

4. If you're using OpenSSH (or another SSH client), you should set the permissions of this file so it is readable only by you.

    On Linux and UNIX, enter the information in the following example.

    ```
    $   chmod 400 id_rsa-gsg-keypair ; ls -l id_rsa-gsg-keypair
    ```

    You receive output similar to the following example.

    ```
    -r--------  1 fred flintstone 1701 Jun 19 17:57 id_rsa-gsg-keypair
    ```

Authorizing Network Access to Your Instances

By default, Amazon EC2 instances do not permit access on any ports. To access your instance with SSH or RDP, your instance must allow incoming traffic on port 22 or 3389, respectively. To open a port for incoming traffic, add a security group rule to a security group that includes your instance. You can use the AWS Management Console or the command line tools (i.e., API tools). If you use the command line tools, use them on your local system, not on the instance itself.

The following instructions authorize incoming SSH or RDP traffic for your instance, but only from your local system's public IP address. If your IP address is dynamic, you must authorize access each time it changes. To allow additional IP address ranges, add a new security group rule for each range.

Before You Get Started

Get the public IP address of your local computer using a service. To locate a service that provides your IP address, use the search phrase "what is my IP address". If you are connecting through an ISP or from behind your firewall without a static IP address, you need to find out the range of IP addresses used by client computers.

AWS Management Console

To add a rule to a security group for SSH access for Linux instances

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Click **Security Groups** in the **Navigation** pane.
    The console displays a list of security groups that belong to the account.

3. Select an EC2 security group that includes your instance.
    Its rules appear on the **Inbound** tab in the lower pane.

4. From the **Create a new rule:** drop-down list, select SSH.

5. In the **Source** field, specify your local system's public IP address in CIDR notation. For example, if your IP address is 203.0.113.0, enter 203.0.113.0/32.
6. Click **Add Rule**.
    An asterisk appears on the **Inbound** tab.

7. Click **Apply Rule Changes**.
    The new rule is created and applied to all instances that belong to the security group.

To add a rule to a security group for RDP access for Windows instances

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Click **Security Groups** in the **Navigation** pane.
   The console displays a list of security groups that belong to the account.
3. Select an EC2 security group that includes your instance.
   Its rules appear on the **Inbound** tab in the lower pane.
4. From the **Create a new rule:** drop-down list, select RDP.

5. In the **Source** field, specify your local system's public IP address in CIDR notation. For example, if your IP address is 203.0.113.0, enter 203.0.113.0/32.
6. Click **Add Rule**.
   An asterisk appears on the **Inbound** tab.
7. Click **Apply Rule Changes**.
   The new rule is created and applied to all instances that belong to the security group.

Command Line Tools

Use the `ec2-authorize` command. For information about the command, see `ec2-authorize` in the *Amazon EC2 Command Line Reference*.

To add a rule to a security group for SSH access

* The following command adds a rule to the default security group that allows incoming traffic on port 22 (SSH port) from your IP address.

```
PROMPT>  ec2-authorize default -p 22 -s your_ip_address/32
GROUP default
PERMISSION default ALLOWS tcp 22 22 FROM CIDR your_ip_address/32
```

To add a rule to a security group for RDP access

* The following command opens port 3389 (RDP port) to your IP address.
  The following example adds a rule to the default security group that allows incoming traffic on port 22 from your IP address.

```
PROMPT>  ec2-authorize default -p 3389 -s your_ip_address/32
GROUP default
PERMISSION default ALLOWS tcp 3389 3389 FROM CIDR your_ip_address/32
```

API

Use the AuthorizeSecurityGroupIngress action.

To authorize access to your instance

1. This first request gives your local system the ability to use SSH (port 22) to connect to any instance in the "default" security group.

```
https://ec2.amazonaws.com/
?Action=AuthorizeSecurityGroupIngress
&GroupName=default
&IpPermissions.1.IpProtocol=tcp
&IpPermissions.1.FromPort=22
&IpPermissions.1.ToPort=22
&IpPermissions.1.IpRanges.1.CidrIp=your-local-system's-public-ip-address/32
&AUTHPARAMS
```

2. This second request gives your local system the ability to use Remote Desktop (port 3389) to connect to any instance in the "default" security group.

```
https://ec2.amazonaws.com/
?Action=AuthorizeSecurityGroupIngress
&GroupName=default
&IpPermissions.1.IpProtocol=tcp
&IpPermissions.1.FromPort=3389
&IpPermissions.1.ToPort=3389
&IpPermissions.1.IpRanges.1.CidrIp=your-local-system's-public-ip-address/32
&AUTHPARAMS
```

3. This third request allows all port 80 traffic into all instances in the "default" security group.

```
https://ec2.amazonaws.com/
?Action=AuthorizeSecurityGroupIngress
&GroupName=default
&IpPermissions.1.IpProtocol=tcp
&IpPermissions.1.FromPort=80
&IpPermissions.1.ToPort=80
&IpPermissions.1.IpRanges.1.CidrIp=0.0.0.0/0
&AUTHPARAMS
```

Connecting to Linux/UNIX Instances Using SSH

Topics

*   Connecting from Your Web Browser Using a Java-Based SSH Client (p. 264)

Connecting from Your Web Browser Using a Java-Based SSH Client

The steps to connect to a Linux/UNIX instance using your browser are:

Install and Enable Java on Your Browser

To connect to your instance from the Amazon Elastic Compute Cloud (Amazon EC2) console, you must have Java installed and enabled in your browser. To install and enable Java, follow the steps Oracle provides below or contact your IT administrator to install and enable Java on your web browser:

Note

On a Windows or Mac client, you must run your Web browser with administrator credentials. For Linux, additional steps may be required if you are not logged in as root.

1. Install Java (see http://java.com/en/download/help/index_installing.xml)
2. Enable Java in your web browser (see http://java.com/en/download/help/enable_browser.xml)

Connect Using a Java-Based (SSH) Client

To connect to your instance through a web browser

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the **Navigation** pane, click **Instances**.
3. Right-click your instance, and then click **Connect**.
4. Click **Connect from your browser using the Java SSH client (Java Required)**. AWS automatically detects the public DNS address of your instance and the key pair name you launched the instance with.
5. In **User name**, enter the user name to log in to your instance.

   ### Note

   For an Amazon Linux instance, the default user name is `ec2-user`. For Ubuntu, the default user name is `ubuntu`. Some AMIs allow you to log in as root.

6. The **Key name** field is automatically populated for you.
7. In **Private key path**, enter the fully qualified path to your .pem private key file.
8. Click **Save key location**, click **Stored in browser cache** to store the key location in your browser cache so the key location is detected in subsequent browser sessions, until your clear your browser's cache.
9. Click **Launch SSH Client**.
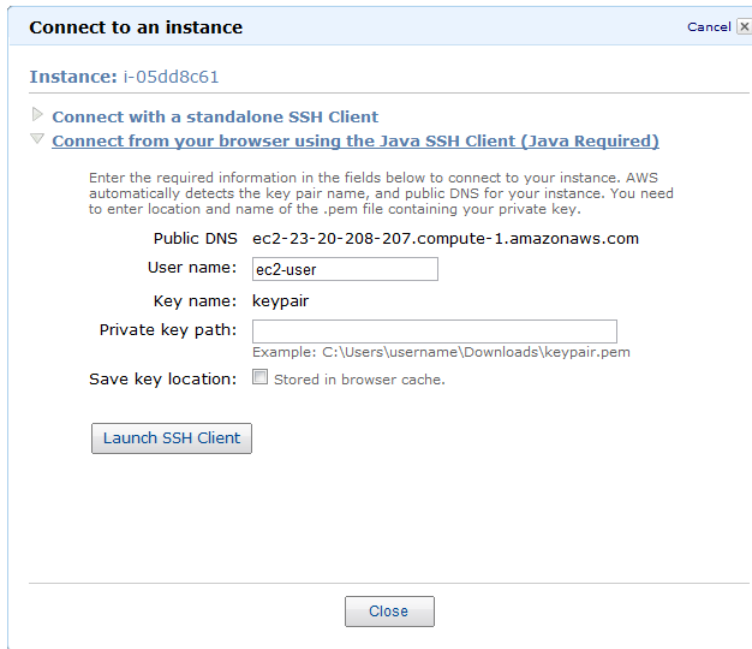
10. If necessary, click **Yes** to trust the certificate.

11. Click **Run** to run the MindTerm client.

12. If you accept the license agreement, click **Accept**.

13. If this is your first time running MindTerm, a series of dialog boxes will ask you to confirm setup for your home directory and other settings.

14. Confirm settings for MindTerm setup. A screen opens and you are connected to your instance.
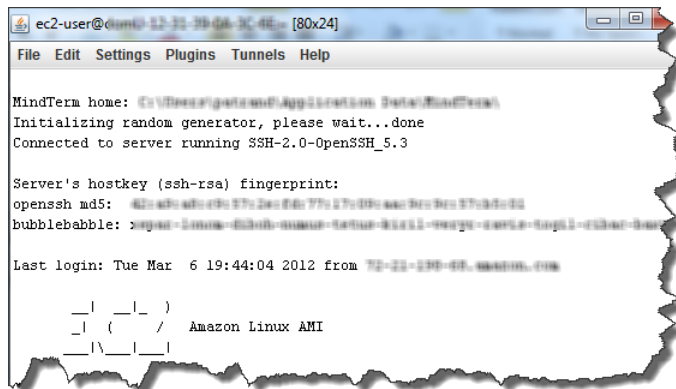
Connect to Linux/UNIX Instances from Linux/UNIX with SSH

This section describes how to connect to Linux and UNIX instances using SSH and SCP on a Linux/UNIX computer.

Prerequisites

- **Enable SSH traffic—**Open SSH port on the instance.

Before you try to connect, ensure that your Amazon EC2 instance accepts incoming SSH traffic (usually on port 22). For more information, see Authorizing Network Access to Your Instances (p. 261).

- Most Linux and UNIX computers include an SSH client by default. You can check for an SSH client by typing `ssh` at the command line. If your computer doesn't recognize the command, the OpenSSH project provides a free implementation of the full suite of SSH tools. For more information, go to http://www.openssh.org.

- **Private key—**Get the path to your private key
  You'll need the fully qualified path of the private key file associated with your instance. For more information on key pairs, see Getting an SSH Key Pair (p. 255).

To use SSH to connect

1. If you've launched a public AMI, run the `ec2-get-console-output` command on your local system (not on the instance), and locate the `SSH HOST KEY FINGERPRINTS` section. For more information, see `ec2-get-console-output` in the *Amazon Elastic Compute Cloud Command Line Reference*.

   ```
   PROMPT>  ec2-get-console-output instance_id


   ...
   ec2: -----BEGIN SSH HOST KEY FINGERPRINTS-----
   ec2: 2048 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
   /etc/ssh/ssh_host_key.pub
   ec2: 2048 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
   /etc/ssh/ssh_host_rsa_key.pub
   ec2: 1024 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
   /etc/ssh/ssh_host_dsa_key.pub
   ec2: -----END SSH HOST KEY FINGERPRINTS-----
   ...
   ```

   Note the fingerprints so that you can compare them to the fingerprints of the instance.

2. In a command line shell, change directories to the location of the private key file that you created when you launched the instance.

3. Use the `chmod` command to make sure your private key file isn't publicly viewable. For example, if your private key file were `My_Keypair.pem`, you would enter:

   ```
   chmod 400 My_Keypair.pem
   ```

4. In the **Navigation** pane, click **Instances**.

5. Right-click your instance, and then click **Connect**.

6. Click **Connect using a standalone SSH client**. AWS automatically detects the public DNS address of your instance and the key pair name you launched the instance with.

7. Copy the example command provided in the Amazon EC2 console if you launched an Amazon Linux instance. If you used a different Amazon Machine Image (AMI) for your Linux/UNIX instance, you need to log in as the default user for the AMI. For an Ubuntu instance, the default user name is `ubuntu`. Some AMIs allow you to log in as `root` so you will need to change the user name from `ec2-user` to the appropriate user name.

   ```
   ssh -i <your key a name>.pem ec2-user@ec2-184-72-204-112.compute-1.amazon
   aws.com
   ```
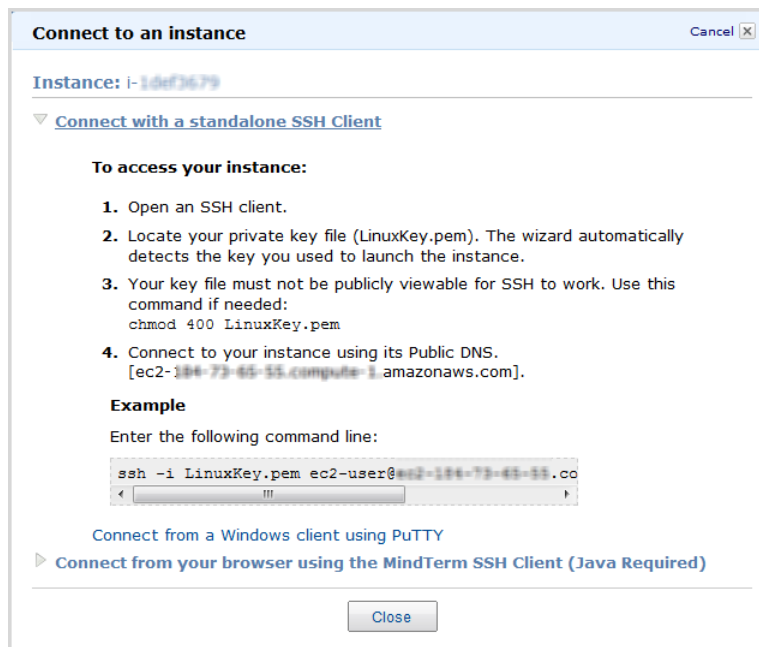
You'll see a response like the following.

```
The authenticity of host 'ec2-184-72-204-112.compute-1.amazonaws.com
(10.254.142.33)'
can't be established.
RSA key fingerprint is 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00.
Are you sure you want to continue connecting (yes/no)? yes
```

> **Important**
>
> If you've launched a public AMI, verify that the fingerprint matches the fingerprint from the
> output of the `ec2-get-console-output` command. If it doesn't, someone might be
> attempting a "man-in-the-middle" attack.

8. Enter **yes**.

   You'll see a response like the following.

```
Warning: Permanently added 'ec2-184-72-204-112.compute-1.amazonaws.com'
(RSA)
    to the list of known hosts.
```

Transfer Files to Linux/UNIX Instances from Linux/UNIX with SCP

One way to transfer files between your local computer and a Linux/UNIX instance is to use Secure Copy
(SCP). This section describes how to transfer files with SCP. The procedure is very similar to the procedure
for connecting to an instance with SSH.

Prerequisites

- **Enable SSH traffic—**Open the instance's SSH port
  Before you try to connect, ensure that your Amazon EC2 instance accepts incoming SSH traffic (usually on port 22). For more information, see Authorizing Network Access to Your Instances (p. 261).

- **SCP client—**Install an SCP client
  Most Linux and UNIX computers include an SCP client by default. If yours doesn't, the OpenSSH project provides a free implementation of the full suite of SSH tools, including an SCP client. For more information, go to http://www.openssh.org.

- **Instance ID—**Get the ID of your Amazon EC2 instance
  Retrieve the Instance ID of the Amazon EC2 instance you want to access. The Instance ID for all your instances are available in the AWS Management Console or through the CLI command ec2-describe-instances.

- **Instance's public DNS—**Get the public DNS of your Amazon EC2 instance
  Retrieve the public DNS of the Amazon EC2 instance you want to access. You can find the public DNS for your instance using the AWS Management Console or by calling the CLI command ec2-describe-instances. The format of an instance's public DNS is ec2-w-x-y-z-compute-1.amazonaws.com where w, x, y, and z each represents a number between 0 and 255 inclusive.

- **Private key—**Get the path to your private key
  You'll need the fully qualified path of the private key file associated with your instance. For more information on key pairs, see Getting an SSH Key Pair (p. 255).

The following procedure steps you through using SCP to transfer a file. If you've already connected to the instance with SSH and have verified its fingerprints, you can start with the step that contains the SCP command (step 4).

To use SCP to transfer a file

1. If you've launched a public AMI, run the `ec2-get-console-output` command on your local system (not on the instance) and locate the `SSH HOST KEY FINGERPRINTS` section. For more information, see `ec2-get-console-output` in the *Amazon Elastic Compute Cloud Command Line Reference*.

```
PROMPT>  ec2-get-console-output instance_id


...
ec2: -----BEGIN SSH HOST KEY FINGERPRINTS-----
ec2: 2048 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
/etc/ssh/ssh_host_key.pub
ec2: 2048 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
/etc/ssh/ssh_host_rsa_key.pub
ec2: 1024 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
/etc/ssh/ssh_host_dsa_key.pub
ec2: -----END SSH HOST KEY FINGERPRINTS-----
...
```

   Note the fingerprints so that you can compare them to the fingerprints of the instance.

2. In a command line shell, change directories to the location of the private key file that you created when you launched the instance.

3. Use the `chmod` command to make sure your private key file isn't publicly viewable. For example, if your file were `My_Keypair.pem`, you would enter:

```
chmod 400 My_Keypair.pem
```

4. Transfer a file to your instance using the instance's public DNS name (available through the AWS Management Console or the `ec2-describe-instances` command). For example, if the key file is `My_Keypair.pem`, the file to transfer is samplefile.txt, and the instance's DNS name is ec2-184-72-204-112.compute-1.amazonaws.com, use the following command to copy the file to the ec2-user home directory.

```
scp -i My_Keypair.pem samplefile.txt ec2-user@ec2-184-72-204-112.compute-
1.amazonaws.com:~
```

   Note

   Some AMIs let you log in as root, but some require that you log in with the username ec2-user. For log in information for your chosen AMI, contact your AMI provider directly or go to Amazon Machine Images(AMIs) page, then locate and click your AMI on the list.

   You'll see a response like the following.

```
The authenticity of host 'ec2-184-72-204-112.compute-1.amazonaws.com
(10.254.142.33)'
can't be established.
RSA key fingerprint is 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00.
Are you sure you want to continue connecting (yes/no)? yes
```

   Important

   If you've launched a public AMI, verify that the fingerprint matches the fingerprint from the output of the `ec2-get-console-output` command. If it doesn't, someone might be attempting a "man-in-the-middle" attack.

5. Enter **yes**.

   You'll see a response like the following.

```
Warning: Permanently added 'ec2-184-72-204-112.compute-1.amazonaws.com'
(RSA)
to the list of known hosts.
Sending file modes: C0644 20 samplefile.txt
Sink: C0644 20 samplefile.txt
samplefile.txt                                      100%   20     0.0KB/s   00:00
```

To transfer files in the other direction, i.e., from your Amazon EC2 instance to your local computer, simply reverse the order of the host parameters. For example, to transfer the samplefile.txt file from your Amazon EC2 instance back to the home directory on your local computer as samplefile2.txt, use the following command on your local computer.

```
scp -i My_Keypair.pem ec2-user@ec2-184-72-204-112.compute-1.amazon
aws.com:~/samplefile.txt ~/samplefile2.txt
```

Connecting to Linux/UNIX Instances from Windows Using PuTTY

Topics

To connect to your Linux/UNIX instance from a Windows computer, use an SSH client. The following instructions explain how to use PuTTY, a free SSH client for Windows computers.

Prerequisites

- **Enable SSH traffic—**Open the instance's SSH port
  Before you try to connect, ensure that your Amazon EC2 instance accepts incoming SSH traffic (usually on port 22). For more information, see Authorizing Network Access to Your Instances (p. 261).

- **Instance ID—**Get the ID of your Amazon EC2 instance
  Retrieve the Instance ID of the Amazon EC2 instance you want to access. The Instance ID for all your instances are available in the AWS Management Console or through the CLI command ec2-describe-instances.

- **Instance's public DNS—**Get the public DNS of your Amazon EC2 instance
  Retrieve the public DNS of the Amazon EC2 instance you want to access. You can find the public DNS for your instance using the AWS Management Console or by calling the CLI command ec2-describe-instances. The format of an instance's public DNS is ec2-w-x-y-z-compute-1.amazonaws.com where w, x, y, and z each represents a number between 0 and 255 inclusive.

- **Private key—**Get the path to your private key
  You'll need the fully qualified path of the private key file associated with your instance. For more information on key pairs, see Getting an SSH Key Pair (p. 255).

Getting PuTTY

To download and install PuTTY

- Go to http://www.chiark.greenend.org.uk/~sgtatham/putty/ and follow the instructions there.

  Note

  Other tools in the PuTTY suite are PuTTYgen, a key generation program, and pscp, a secure copy command line tool. The different PuTTY tools are separate applications. You can install them separately or install the entire suite with a simple Windows installer. The following instructions assume you've installed the entire suite and can access all the components from the Windows Start menu.

Converting Your Private Key

PuTTY does not natively support the private key format generated by Amazon EC2. Fortunately, PuTTY has a tool called PuTTYgen, which can convert keys to the required PuTTY format.

To convert your private key

1. Start PuTTYgen (e.g., from the **Start** menu, click **All Programs > PuTTY > PuTTYgen**).
2. Click **Load** and browse to the location of the private key file that you want to convert (e.g., GSG_Keypair.pem). By default, PuTTYgen displays only files with extension .ppk; you'll need to

change that to display files of all types in order to see your `.pem` key file. The private key file must end with a newline character or PuTTYgen cannot load it correctly.

3. Select your `.pem` key file and click **Open**.

   PuTTYgen displays the following message.
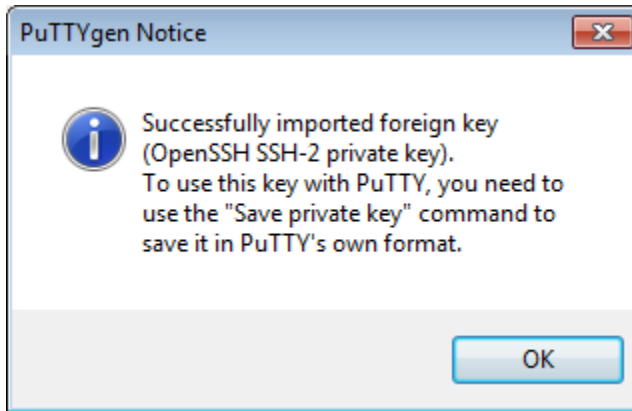
   When you click OK, PuTTYgen displays a dialog box with information about the key you loaded, such as the public key and the fingerprint. The keys that Amazon EC2 generates are 1024-bit SSH-2 RSA keys.

4. Click **Save private key** to save the key in PuTTY's format.
   PuTTYgen asks if you want to save the key without a passphrase.
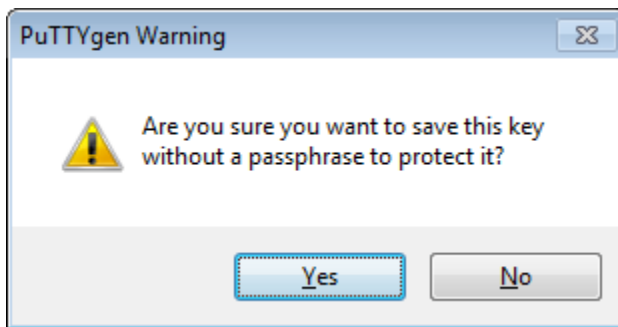
5. Click **Yes**.

   Note

   A passphrase on a private key is an extra layer of protection, so even if your private key is discovered, it will not be usable without the passphrase. The downside to using a passphrase is that it makes automation harder because human intervention is needed to log on to an instance, or copy files to an instance. For this exercise, we're not using a passphrase.

6. Use the same name for the key that you used for the key pair (for example, GSG_Keypair). PuTTY automatically adds the `.ppk` file extension.

Your private key is now in the correct format for use with PuTTY. You can now connect to your instance using PuTTY's SSH client.

Connecting Using PuTTY SSH

You'll connect by starting a PuTTY SSH session.

To use SSH to connect

1. If you've launched a public AMI that isn't one that you created yourself, run the
    `ec2-get-console-output` command on your local system (not on the instance), and locate the
    `SSH HOST KEY FINGERPRINTS` section. For more information, see `ec2-get-console-output`
    in the *Amazon Elastic Compute Cloud Command Line Reference*.

    ```
    PROMPT>  ec2-get-console-output instance_id

    ...
    ec2: -----BEGIN SSH HOST KEY FINGERPRINTS-----
    ec2: 2048 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
    /etc/ssh/ssh_host_key.pub
    ec2: 2048 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
    /etc/ssh/ssh_host_rsa_key.pub
    ec2: 1024 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
    /etc/ssh/ssh_host_dsa_key.pub
    ec2: -----END SSH HOST KEY FINGERPRINTS-----
    ...
    ```

    Note the fingerprints so that you can compare them to the fingerprints of the instance.

2. Start PuTTY (from the **Start** menu, click **All Programs > PuTTY > PuTTY**).
    A dialog box opens with a **Category** menu on the left side. On the right side, the basic options for
    your PuTTY session are displayed.

3. In the **Host Name** field, enter the public DNS name of your instance (available through the AWS
    Management Console or the `ec2-describe-instances` command). You can optionally prefix the
    DNS name with `ec2-user@` to automatically log in with superuser privileges when the session opens.
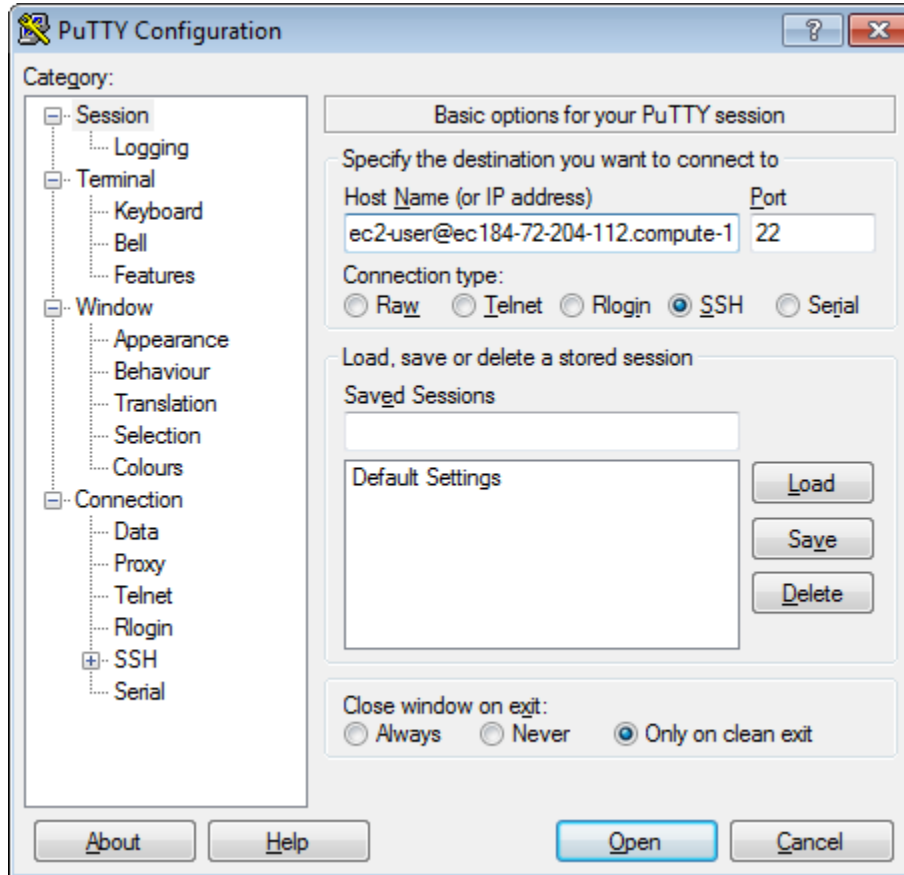
4. In the **Category** menu, under **Connection**, click **SSH**, and then **Auth**.
   The options controlling SSH authentication are displayed.

5. Click **Browse** and navigate to the PuTTY private key file you generated in the preceding section.
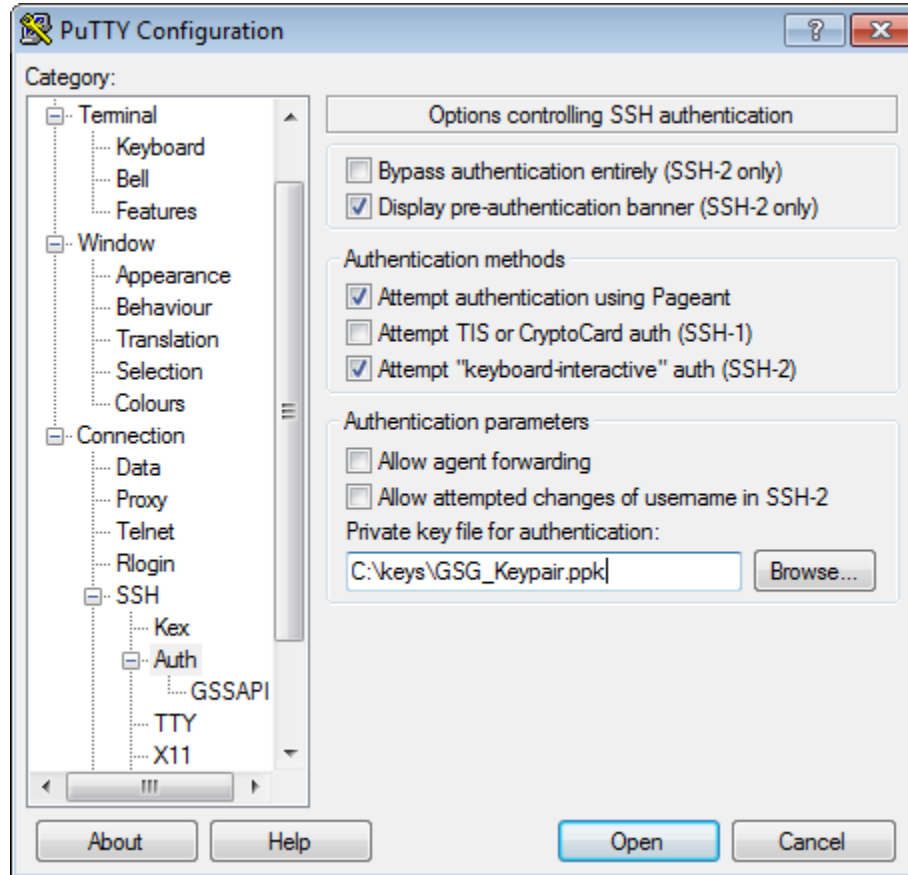
6. Click **Open**.
   An SSH session window opens and PuTTY displays a security alert asking if you trust the host you're connecting to.

   Important

   If you've launched a public AMI, verify that the fingerprint in the security alert matches the fingerprint from the output of the `ec2-get-console-output` command. If it doesn't, someone might be attempting a "man-in-the-middle" attack.

7. Click **Yes**.
8. In the SSH session window, log in as root (or ec2-user) if you didn't as part of starting the SSH session.

   Note

   Some AMIs let you log in as root, but some require you to log in with the username ec2-user. For log in information for your chosen AMI, contact your AMI provider directly or go to Amazon Machine Images(AMIs) page, then locate and click your AMI on the list.

   Note

   If you specified a passphrase when you converted your private key to PuTTY's format, you must provide that passphrase when you log in to the instance.

Transferring Files with PSCP

The PuTTY Secure Copy Client (PSCP) is a command-line tool that lets you transfer files between your Windows computer and your Linux/UNIX instance.

To use PSCP, you'll need the private key you generated in Converting Your Private Key (p. 270). You'll also need the public IP address of your Linux/UNIX instance.

The following example transfers the file `sample_file.txt` from a Windows computer to the `/usr/local` directory on a Linux/UNIX instance:

```
Prompt>pscp -i C:\GSG_Keypair.ppk C:\sample_file.txt root@ec2-184-72-204-
112.compute-1.amazonaws.com:/usr/local/sample_file.txt
```

If you prefer a graphical user interface (GUI), you can use an open source GUI tool named WinSCP. For more information, go to the WinSCP website.

Connecting to Windows Instances

Topics
- Connect to Windows Instances with RDP (p. 275)
- Transfer Files to Windows Instances from Windows (p. 277)

This section describes how to connect to instances running Windows from local computers running Windows, Linux/UNIX, or Mac OS.

Prerequisites

- **Enable RDP traffic—**Open the instance's RDP port
  Before you try to connect, ensure that your Amazon EC2 instance accepts incoming RDP traffic (usually on port 3389). For more information, see Authorizing Network Access to Your Instances (p. 261).

- **RDP client—**Install an RDP client
  Windows computers include an RDP client by default. You can check for an RDP client by typing **mstsc** at the command line. If your computer doesn't recognize this command, go to the Microsoft Windows home page and search for the download for Remote Desktop Connection. For Mac OS X, you can use Microsoft's Remote Desktop Client. For Linux/UNIX, you can use rdesktop.

- **Instance ID—**Get the ID of your Amazon EC2 instance
  Retrieve the Instance ID of the Amazon EC2 instance you want to access. The Instance ID for all your instances are available in the AWS Management Console or through the CLI command ec2-describe-instances.

- **Private key—**Get the path to your private key
  You'll need the fully qualified path of the private key file associated with your instance. For more information on key pairs, see Getting an SSH Key Pair (p. 255).

Connect to Windows Instances with RDP

To connect to a Windows instance, you must retrieve the initial administrator password first, and then use it with Remote Desktop. The password is encrypted using the key pair that you provided and stored within the <password> tags of the console output. Therefore, you'll need the contents of the private key file that you created when you launched the instance (for example, `GSG_Keypair.pem`).

To connect to your Windows instance

1. If you've launched a public AMI, get the instance's RDP certificate.

    a. Go to the Amazon EC2 console and locate the instance on the **Instances** page.

    b. Right-click the instance and select **Get System Log**.
        The **System Log** dialog box is displayed (it might take a few minutes after the instance is launched before the RDP certificate is available).

        A thumbprint is a series of hexadecimal numbers enclosed in a `<THUMBPRINT>` tag. For example, a thumbprint might look like
        `<THUMBPRINT>2C81502A74D7B112DF801E460A16F53DFEXAMPLE</THUMBPRINT>`. Note the displayed thumbprint; we'll compare this thumbprint to the thumbprint of the instance as we connect to it.

2. Retrieve the initial administrator password:

    a. Go to the Amazon EC2 console and locate the instance on the **Instances** page.

    b. Right-click the instance and select **Get Windows Password** (it might take a few minutes after the instance is launched before the password is available).
        The **Retrieve Default Windows Administrator Password** dialog box is displayed.

    c. Click **Browse** and navigate to the private key file you created when you launched the instance. Select the file and click **OK** to paste the contents of the private key file into the **Private Key** field.

    d. Click **Decrypt Password**.
        The console returns the default administrator password for the instance.

    e. Save the password. You'll need it to connect to the instance.

3. Connect to the instance using Remote Desktop:

    a. Go to the Amazon EC2 console and locate the instance on the **Instances** page.

    b. Right-click the instance and select **Connect**.

    c. Click **Download shortcut file**.
        Your browser prompts you to either open or save the .rdp file. Either option is fine. When you have finished, click **Close**.

    d. Launch the shortcut file.

    e. Log in to the instance as prompted, using `Administrator` as the user name and the default administrator password you saved in the previous step.
        The Amazon EC2 instance returns a security alert.

    f. To verify the instance, click **View Certificate**.
        The **Certificate** page appears.

    g. Click the **Details** tab.
        The **Details** page appears.

    h. Select **Thumbprint** and verify its value against the value you wrote down previously.

> **Important**
>
> If you've launched a public AMI, verify that the thumbprint matches a thumbprint from the instance's RDP certificate. If it doesn't, someone might be attempting a "man-in-the-middle" attack.

i.   If the thumbprints match, click **OK** and then **Yes**.
The Remote Desktop Connection client connects to the instance.

Transfer Files to Windows Instances from Windows

One way to transfer files between an Amazon EC2 Windows instance and your local Windows computer is to use the local file sharing feature of Windows Remote Desktop. If you enable this option in your Windows Remote Desktop Connection software, you can access your local files from your Amazon EC2 Windows instances. You can access local files on hard disk drives, DVD drives, portable media drives, and mapped network drives.

For information about this feature, go to the Microsoft Support website or go to The most useful feature of Remote Desktop I never knew about on the MSDN Blogs website.

Managing Instances

Topics

- Auto-Scaling and Load Balancing Your Instances (p. 277)
- Monitoring Your Instances (p. 279)
- Ensuring Idempotency (p. 300)
- Instance Metadata (p. 302)

This section describes features that help you manage your instances.

Auto-Scaling and Load Balancing Your Instances

If your expect your application to have significant variability in usage, you might want to use Auto Scaling and Elastic Load Balancing, two features of Amazon EC2 that help manage the variability.

Auto Scaling

Auto Scaling enables you to scale up or down the number of instances you are using based on parameters that you specify, such as traffic or CPU load.

Auto Scaling also monitors the health of each Amazon EC2 instance that it launches. If any instance terminates unexpectedly, Auto Scaling detects the termination and launches a replacement instance.

For a high degree of flexibility, you can organize Amazon EC2 instances into AutoScalingGroups, which enable you to scale different server classes (e.g., web servers, back end servers) at different rates. For each group, you specify the minimum number of instances, the maximum number of instances, and the parameters to increase and decrease the number of running instances.

For information on setting up Auto Scaling, go to the Amazon Auto Scaling Developer Guide.

Load Balancing

Elastic Load Balancing lets you automatically distribute the incoming traffic (or load) among all the instances you are running. The service also makes it easy to add new instances when you need to increase the capacity of your web site application.

Customers reach your web site via your web URL, such as www.mywebsite.com. This single address might actually represent several instances of your running web application. To always have an available web site, you need to run multiple instances. Otherwise, your customers might see delays when accessing your site, or worse, might not be able to access your site at all.

Elastic Load Balancing manages the incoming requests by optimally routing traffic so that no one instance is overwhelmed. You can quickly add more instances to applications that are experiencing an upsurge in traffic or remove capacity when traffic is slow.

For information on setting up Elastic Load Balancing, go to the Elastic Load Balancing Developer Guide.

Monitoring Your Instances

Topics

Amazon Web Services (AWS) automatically provides data, such as Amazon CloudWatch metrics and instance status and volume status, that you can use to monitor your Amazon EC2 instances and Amazon Elastic Block Store (Amazon EBS) volumes:

- CloudWatch metrics are statistical data you can use to view, analyze, and set alarms on the operational behavior of your instances and volumes. These metrics include CPU utilization, network traffic, I/O, and latency.
- Instance status provides two types of information:
  - Instance status checks that summarize results of automated tests that you can use to determine whether your instances are affected by specific, detectable problems.
  - Events that provide information about certain activities that are scheduled for your instances, including operational maintenance that AWS may perform such as rebooting and retirement.
- Volume status provides information about the IO (read/write) capability of your Amazon Elastic Block Store (Amazon EBS) volume. It is designed to provide you the information you need in order to understand when your EBS volumes are impaired and what actions to consider taking.

Monitoring Your Instances and Volumes with CloudWatch

Topics

Amazon CloudWatch is a service that collects raw data from partnered AWS products such as Amazon EC2 and then processes the information into readable, near real-time metrics. These statistics are recorded for a period of two weeks, allowing you access to historical information and providing you with a better perspective on how your web application or service is performing. For detailed information about Amazon CloudWatch, go to the Amazon CloudWatch Developer Guide.

The following table describes the types of Amazon EC2 monitoring data available to you.

Resource	Type	Description
Instances	Basic	Data is available automatically in 5-minute periods at no charge.
	Detailed	Data is available in 1-minute periods at an additional cost. To get this level of data, you must specifically enable it for the instance. For the instances where you've enabled detailed monitoring, you can also get aggregated data across groups of similar instances. For information about pricing, go to the Amazon CloudWatch product page.
Volumes	Basic	Data is available automatically in 5-minute periods at no charge. No detailed data is available for Amazon EBS volumes.

Monitoring Instances

You can get monitoring data for your Amazon EC2 instances using either the Amazon CloudWatch API or the AWS Management Console. The console displays a series of graphs based on the raw data from the Amazon CloudWatch API. Depending on your needs, you might prefer to use either the data from the API or the graphs in the console.

Data from the Amazon Cloudwatch API

You can use the Amazon CloudWatch `GetMetricStatistics` API to get any of the instance metrics listed in the following table. The *period* refers to how often the system reports a data point for each metric for an instance. If you've enabled detailed monitoring, each data point covers the instance's previous 1 minute of activity. Otherwise, each data point covers the instance's previous 5 minutes of activity.

Metric	Description
`CPUUtilization`	The percentage of allocated EC2 compute units that are currently in use on the instance. This metric identifies the processing power required to run an application upon a selected instance. Units: *Percent*

Metric	Description
DiskReadOps	Completed read operations from all ephemeral disks available to the instance (if your instance uses Amazon EBS, see Amazon EBS Metrics (p. 290).)
	This metric identifies the rate at which an application reads a disk. This can be used to determine the speed in which an application reads data from a hard disk.
	Units: *Count*
DiskWriteOps	Completed write operations to all ephemeral disks available to the instance (if your instance uses Amazon EBS, see Amazon EBS Metrics (p. 290).)
	This metric identifies the rate at which an application writes to a hard disk. This can be used to determine the speed in which an application saves data to a hard disk.
	Units: *Count*
DiskReadBytes	Bytes read from all ephemeral disks available to the instance (if your instance uses Amazon EBS, see Amazon EBS Metrics (p. 290).)
	This metric is used to determine the volume of the data the application reads from the hard disk of the instance. This can be used to determine the speed of the application.
	Units: *Bytes*
DiskWriteBytes	Bytes written to all ephemeral disks available to the instance (if your instance uses Amazon EBS, see Amazon EBS Metrics (p. 290).)
	This metric is used to determine the volume of the data the application writes onto the hard disk of the instance. This can be used to determine the speed of the application.
	Units: *Bytes*
NetworkIn	The number of bytes received on all network interfaces by the instance. This metric identifies the volume of incoming network traffic to an application on a single instance.
	Units: *Bytes*
NetworkOut	The number of bytes sent out on all network interfaces by the instance. This metric identifies the volume of outgoing network traffic to an application on a single instance.
	Units: *Bytes*

Metric	Description
StatusCheckFailed	A combination of StatusCheckFailed_Instance and StatusCheckFailed_System that reports if either of the status checks has failed. Values for this metric are either 0 (zero) or 1 (one.) A zero indicates that the status check passed. A one indicates a status check failure. **Note** Status check metrics are available at 5 minute frequency and are not available in Detailed Monitoring. For a newly launched instance, status check metric data will only be available after the instance has completed the initialization state. Status check metrics will become available within a few minutes of being in the running state. Units: *Count*
StatusCheckFailed_Instance	Reports whether the instance has passed the EC2 instance status check in the last 5 minutes. Values for this metric are either 0 (zero) or 1 (one.) A zero indicates that the status check passed. A one indicates a status check failure. **Note** Status check metrics are available at 5 minute frequency and are not available in Detailed Monitoring. For a newly launched instance, status check metric data will only be available after the instance has completed the initialization state. Status check metrics will become available within a few minutes of being in the running state. Units: *Count*
StatusCheckFailed_System	Reports whether the instance has passed the EC2 system status check in the last 5 minutes. Values for this metric are either 0 (zero) or 1 (one.) A zero indicates that the status check passed. A one indicates a status check failure. **Note** Status check metrics are available at 5 minute frequency and are not available in Detailed Monitoring. For a newly launched instance, status check metric data will only be available after the instance has completed the initialization state. Status check metrics will become available within a few minutes of being in the running state. Units: *Count*

Note

When you get data from the Amazon CloudWatch system, you can specify a *Period* request parameter (i.e., how granular you want the returned data to be). This is different than the period we use when we collect the data (either 1-minute periods for detailed monitoring, or 5-minute periods for basic monitoring). We recommend you specify a period in your request that is equal to or larger than the collection period to ensure the returned data is valid.

You can use the dimensions in the following table to refine the metrics returned for your instances.

Dimension	Description
AutoScalingGroupName	This dimension filters the data you request for all instances in a specified capacity group. An *AutoScalingGroup* is a collection of instances you define if you're using the Auto Scaling service. This dimension is available only for EC2 metrics when the instances are in such an AutoScalingGroup. Available for instances with Detailed or Basic Monitoring enabled.
ImageId	This dimension filters the data you request for all instances running this EC2 Amazon Machine Image (AMI). Available for instances with Detailed Monitoring enabled.
InstanceId	This dimension filters the data you request for the identified instance only. This helps you pinpoint an exact instance from which to monitor data. Available for instances with Detailed Monitoring enabled.
InstanceType	This dimension filters the data you request for all instances running with this specified instance type. This helps you categorize your data by the type of instance running. For example, you might compare data from an m1.small instance and an m1.large instance to determine which has the better business value for your application. Available for instances with Detailed Monitoring enabled.

For more information about using the `GetMetricStatistics` action, go to GetMetricStatistics in the *Amazon CloudWatch API Reference*.

Graphs in the AWS Management Console

After you launch an instance, you can go to the AWS Management Console and view the instance's monitoring graphs. They're displayed when you select the instance on the **Instances** page in the EC2 Dashboard. A **Monitoring** tab is displayed next to the instance's **Description** tab. The following graphs are available:

- Average CPU Utilization (Percent)
- Average Disk Reads (Bytes)
- Average Disk Writes (Bytes)
- Maximum Network In (Bytes)
- Maximum Network Out (Bytes)
- Summary Disk Read Operations (Count)
- Summary Disk Write Operations (Count)
- Summary Status (Any)
- Summary Status Instance (Count)
- Summary Status System (Count)

The AWS Management Console contains a console for Amazon CloudWatch. In the Amazon CloudWatch console you can search and browse all your AWS resource metrics, view graphs to troubleshoot issues and discover trends, create and edit alarms to be notified of problems, and see at-a-glance overviews of your alarms and AWS resources. For more information, go to AWS Management Console in the *Amazon CloudWatch Developer Guide*.

Enabling or Disabling Detailed Monitoring on an Amazon EC2 Instance

This section describes how to enable or disable detailed monitoring on either a new instance (as you launch it) or on a running or stopped instance. After you enable detailed monitoring, the Amazon EC2 console in the AWS Management Console displays monitoring graphs with a 1-minute period for the instance. You can enable or disable detailed monitoring using the console or the command line interface (CLI).

AWS Management Console

To enable detailed monitoring of an existing EC2 instance

You can enable detailed monitoring of your EC2 instances, which provides data about your instance in 1-minute periods. (There is an additional charge for 1-minute monitoring.) Detailed data is then available for the instance in the AWS Management Console graphs or through the API. To get this level of data, you must specifically enable it for the instance. For the instances on which you've enabled detailed monitoring, you can also get aggregated data across groups of similar instances. An instance must be running or stopped to enable detailed monitoring.

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the **Navigation** pane, click **Instances**.
3. In **My Instances**, select a running or stopped instance, click **Instance Actions**, and then click **Enable Detailed Monitoring**.
4. In the **Enable Detailed Monitoring** dialog box, click **Yes, Enable**.
5. In the **Enable Detailed Monitoring** confirmation dialog box, click **Close**.

   Detailed data (collected with a 1-minute period) is then available for the instance in the AWS Management Console graphs or through the API.

To enable detailed monitoring when launching an EC2 instance

- When launching an instance with the AWS Management Console, select the check box for **Enable detailed CloudWatch Monitoring for this instance** on the **Advanced Instance Options** section of the Classic launch wizard.

  After the instance is launched, you can select the instance in the console and view its monitoring graphs on the instance's **Monitoring** tab in the lower pane.

To disable detailed monitoring of an EC2 instance

When you no longer want to monitor your instances at 1-minute intervals, you can disable detailed monitoring and use basic monitoring instead. Basic monitoring provides data in 5-minute periods at no charge.

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the **Navigation** pane, click **Instances**.
3. In **My Instances**, select an instance, click **Instance Actions**, and then click **Disable Detailed Monitoring**.
4. In the **Disable Detailed Monitoring** dialog box, click **Yes, Disable**.
5. In the **Disable Detailed Monitoring** confirmation dialog box, click **Close**.

For information about launching instances, see Launching Amazon EC2 Instances (p. 217).

Command Line Interface

To enable detailed monitoring on an existing instance

- Use the `ec2-monitor-instances` command with one or more instance IDs.

```
PROMPT>  ec2-monitor-instances i-1a2b3c4d
i-1a2b3c4d    monitoring-pending
```

Detailed data (collected with a 1-minute period) is then available for the instance in the AWS Management Console graphs or through the API.

To enable detailed monitoring when launching an instance

- Use the `ec2-run-instances` command with the `--monitor` flag.

```
PROMPT>  ec2-run-instances ami-2bb65342 -k gsg-keypair --monitor
```

Amazon EC2 returns output similar to the following example. The status of monitoring is listed as *monitoring-enabled*.

```
RESERVATION    r-7430c31d      111122223333    default
INSTANCE       i-ae0bf0c7      ami-2bb65342    pending gsg-keypair    0
 m1.small    2008-03-21T16:19:25+0000      us-east-1a    monitoring-enabled
```

After the instance is running, detailed data (collected with a 1-minute period) is then available for the instance in the AWS Management Console graphs or through the API.

To disable detailed monitoring of an instance

- Use the `ec2-unmonitor-instances` command with one or more instance IDs.

```
PROMPT>  ec2-unmonitor-instances i-1a2b3c4d
```

Create a CloudWatch Alarm

You can create an Amazon CloudWatch alarm that monitors any one of your Amazon EC2 instance's CloudWatch metrics. CloudWatch will automatically send you a notification when the metric reaches a threshold you specify. You can create a CloudWatch alarm on the Amazon EC2 console of the AWS Management Console, or you can use the CloudWatch console and configure more advanced options.

To create a CloudWatch alarm for high CPU

1. On the Amazon EC2 console, select the instance for which you want to create an alarm.
2. On the instance's **Monitoring** tab in the lower pane, click **Create Alarm**.
3. In the **Create Alarm** dialog box, set the criteria for your alarm. In this example, we'll set an alarm if the instance's average CPU utilization is above 70 percent.
4. The check box next to Send a notification is selected by default. Click **Create topic**, and either use the default name, or select an existing name. (Notifications use Amazon Simple Notification Service (Amazon SNS)).

5. In the **With these recipients** box, enter the email addresses of the recipients you want to notify. You can enter up to 10 email addresses, each separated by a comma.

6. Configure the threshold for your alarm.

   a. In the **Whenever** boxes, select **Average** and **CPU Utilization**.

   b. In the **Is** boxes, define the threshold for the alarm by selecting **>** and entering **70**.

   c. In the **For at least** boxes, specify the sampling period and number of samples evaluated by the alarm. You can leave the defaults or define your own. For our example, we'll monitor for 1 period of 15 minutes.

      Note

      A shorter period creates a more sensitive alarm. A longer period can mitigate brief spikes in a metric.

   d. In **Alarm name**, a name is automatically generated for you. Click **Edit** if you want to change the name.

      Important

      You cannot modify the name after you create the alarm.

7. Click **Create Alarm**.

After you create the alarm, you can use the **Monitoring** tab in the Amazon EC2 console to view a summary of alarms that have been set for that instance. From there, you can also edit the alarm.

   Note

   If you created a new Amazon SNS topic for this alarm or added new email addresses to an existing topic, each email address added will receive a subscription confirmation email from

Amazon SNS. The person who receives the email must confirm it by clicking the included link in order to receive notifications.

Creating and Editing Status Check Alarms

You can create instance status and system status alarms to notify you when an instance has a failed status check. To create or change these alarms you can use either the AWS Management Console or the command line interface (CLI).

AWS Management Console

To create a status check alarm

You can create status check alarms for an existing instance to monitor instance status or system status. You can configure the alarm to send you a notification by email when an instance fails an instance check or system status check.

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the **Navigation** pane, click **Instances**.
3. In **My Instances**, select an instance, and then on the **Status Checks** tab, click **Create Status Check Alarm**.
4. In the **Create Alarm for** dialog box, select the **Send a notification to** check box, and then choose an existing Amazon Simple Notification Service (SNS) topic or create a new SNS topic to use for this alarm.

5. In the **With these recipients** box, type your email address (e.g., john.stiles@example.com) and the addresses of any additional recipients, separated by commas.
6. In the **Whenever** drop-down list, select the status check you want to be notified about (e.g., Status Check Failed (Any), Status Check Failed (Instance), or Status Check Failed (System)).
7. In the **For at least** box, set the number of periods you want to evaluate (e.g., 2) and in the **consecutive periods** drop-down menu, select the evaluation period duration (e.g., 5 minutes) before triggering the alarm and sending an email

8. To change the default name for the alarm, in the **Name this alarm** box, type a friendly name for the alarm (e.g., StatusCheckFailed), and then click **Create Alarm**.

> **Important**
>
> If you added an email address to the list of recipients or created a new topic, Amazon SNS will send a subscription confirmation email message to each new address shortly after you create an alarm. Remember to click the link contained in that message, which confirms your subscription. Alert notifications are only sent to confirmed addresses.

To edit a status check alarm

If you need to make changes to an instance status alarm, you can edit it.

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the **Navigation** pane, click **Instances**.
3. In **My Instances**, select an instance, click **Instance Actions**, and then click **Add/Edit Alarms**.
4. In the **Alarm Details** dialog box, click the name of the alarm.
5. In the **Edit Alarm for** dialog box, make the desired changes, and then click **Save Alarm**.
6. In the **Alarm Saved Successfully** dialog box, click **Close**.

Command Line Interface

To create a status check alarm using the CLI

You can create a status check alarm using the Amazon CloudWatch CLI. In the following example, the alarm publishes a notification to a specific SNS topic that has the ARN arn:aws:sns:us-east-1:1111111111:StatusCheckNotifications when instance i-ab12345 fails either the instance check or system status check for at least two periods. (The metric is StatusCheckFailed.) For more information, see the mon-put-metric-alarm command in the *Amazon CloudWatch Developer Guide*.

```
mon-put-metric-alarm  --alarm-name StatusCheckFailed-Alarm-for-i-ab12345
    --alarm-description "Alarm when StatusCheckFailed metric has a value of one
 for two
    periods" --metric-name StatusCheckFailed --namespace AWS/EC2 --statistic
Maximum
    --period 300 --threshold 1 --comparison-operator GreaterThanThreshold  --
dimensions
    "InstanceId=i-ab12345"  --evaluation-periods 2 --alarm-actions
    arn:aws:sns:us-east-1:1111111111StatusCheckNotifications  --unit Count
```

Where:

The **--alarm-name** is the name of the alarm. This is required.

The **--alarm-description** is a friendly description of the alarm.

The **--metric-name** is one of the available status metrics (e.g., StatusCheckFailed, StatusCheckFailed_Instance, or StatusCheckFailed_System). This is required.

The **--statistic** is one of the following values: Average, Sum, Minimum, or Maximum. This is required.

The -–**period** is the time frame (in seconds) in which Amazon CloudWatch metrics are collected. In this example, you would enter 300, which is 60 seconds multiplied by 5 minutes. This is required.

The **--threshold** is the value to which the metric will be compared (e.g., 1). This is required.

The **--namespace** is the metric's namespace (e.g., AWS/EC2). This is required.

The **--dimensions** are associated with the metric (e.g., InstanceId=i-ab12345).

The **--evaluation-periods** is the number of consecutive periods for which the value of the metric must be compared to the threshold. This is required.

The **--alarm-actions** is the list of actions to perform when this alarm is triggered. Each action is specified as an Amazon Resource Number (ARN). In this example, we want the alarm to send us an email using Amazon SNS.

Note

You can find the ARN for the Amazon SNS topic that the alarm will use in the Amazon SNS console:

1. Open the Amazon SNS console at https://console.aws.amazon.com/sns/.
2. On the **Navigation** pane, under **My Topics**, select the topic you want the alarm to send mail to.
3. The ARN is located in the **Topic ARN** field on the **Topic Details** pane.

The --**unit** is the unit of the metric on which to alarm (e.g., Count).

1. At a command prompt, type **mon-list-metrics --headers** to view the list of all available Amazon CloudWatch metrics for the services in AWS that you're using.
2. In the list of metrics, look in the **Namespace** column (second column), and review the Status Check metrics that have the AWS/EC2 namespace. These are the status check metrics that you can use to create a status check alarm.
3. At the command prompt, enter the following command:

Monitoring Amazon EBS Volumes

The following table describes the types of monitoring data available for your Amazon EBS volumes.

Type	Description
Basic	Data is available automatically in 5-minute periods at no charge. This includes data for the root device volumes for Amazon EBS-backed instances.
	No detailed data is available for Amazon EBS volumes.

Note

When you get data from Amazon CloudWatch, you can specify a `Period` request parameter (i.e., how granular you want the returned data to be). This is different than the period we use when we collect the data (5-minute periods). We recommend you specify a period in your request that is equal to or larger than the collection period to ensure the returned data is valid.

You can get the data using either the Amazon CloudWatch API or the AWS Management Console. The console takes the raw data from the Amazon CloudWatch API and displays a series of graphs based on the data. Depending on your needs, you might prefer to use either the data from the API or the graphs in the console.

Amazon EBS Metrics

You can use the Amazon CloudWatch `GetMetricStatistics` API to get any of the Amazon EBS volume metrics listed in the following table. The period for all the metrics is 5 minutes, which means the system reports one data point every 5 minutes for each metric for each volume, and that data point covers the volume's previous 5 minutes of activity.

The following table groups metrics that are similar. The metrics in the first two rows are also available for the local stores on Amazon EC2 instances.

Metric	Description

`VolumeReadBytes`
 	
`VolumeWriteBytes`
	The total number of bytes transferred in the period.
	
	Units: Bytes

`VolumeReadOps`
 	
`VolumeWriteOps`
	The total number of operations in the period.
	
	Units: Count

`VolumeTotalReadTime`
 	
`VolumeTotalWriteTime`
	The total number of seconds spent by all operations that completed in the period. If multiple requests are submitted at the same time, this total could be greater than the length of the period. For example, say the period is 5 minutes (300 seconds); if 700 operations completed during that period, and each operation took 1 second, the value would be 700 seconds.
	
	Units: Seconds

`VolumeIdleTime`	The total number of seconds in the period when no read or write operations were submitted. Units: Seconds
`VolumeQueueLength`	The number of read and write operation requests waiting to be completed in the period. Units: Count

Graphs in the AWS Management Console

Once you create a volume, you can go to the Amazon EC2 console and view the volume's monitoring graphs. They're displayed when you select the volume on the **Volumes** page in the EC2 console. A **Monitoring** tab is displayed next to the volume's **Description** tab. The following table lists the graphs that are displayed. The column on the right describes how the raw data metrics from the Amazon CloudWatch API are used to produce each graph. The period for all the graphs is 5 minutes.

Graph Name	Description Using Raw Metrics
Read Bandwidth (KiB/s)	Sum(VolumeReadBytes) / Period / 1024
Write Bandwidth (KiB/s)	Sum(VolumeWriteBytes) / Period / 1024
Read Throughput (Ops/s)	Sum(VolumeReadOps) / Period
Write Throughput (Ops/s)	Sum(VolumeWriteOps) / Period
Avg Queue Length (ops)	Avg(VolumeQueueLength)

Graph Name	Description Using Raw Metrics
% Time Spent Idle	Sum(VolumeIdleTime) / Period * 100
Avg Read Size (KiB/op)	Avg(VolumeReadBytes) / 1024
Avg Write Size (KiB/op)	Avg(VolumeWriteBytes) / 1024
Avg Read Latency (ms/op)	Avg(VolumeTotalReadTime) * 1000
Avg Write Latency (ms/op)	Avg(VolumeTotalWriteTime) * 1000

For the average latency graphs and average size graphs, the average is calculated over the total number of operations (read or write, whichever is applicable to the graph) that completed during the period.

The AWS Management Console contains a console for Amazon CloudWatch. In the Amazon CloudWatch console you can search and browse all your AWS resource metrics, view graphs to troubleshoot issues and discover trends, create and edit alarms to be notified of problems, and see at-a-glance overviews of your alarms and AWS resources. For more information, go to AWS Management Console in the *Amazon CloudWatch Developer Guide*.

Monitoring the Status of Your Instances

You can monitor the status of your instances by viewing status checks and scheduled events for your instances.

A status check gives you the information that results from automated checks performed by Amazon EC2. These automated checks detect whether specific issues are affecting your instances. The status check information, together with the data provided by Amazon CloudWatch, gives you detailed operational visibility into each of your instances.

You can also see status on specific events scheduled for your instances. Events provide information about upcoming activities such as rebooting or retirement that are planned for your instances, along with the scheduled start and end time of each event.

Topics

Monitoring Instances with Status Checks

With instance status monitoring you can quickly determine whether Amazon EC2 has detected any problems that may prevent your instances from running applications. Amazon EC2 performs automated checks on every running Amazon EC2 instance to identify hardware and software issues. You can view the results of these status checks to identify specific and detectable problems. This data augments the information that Amazon EC2 already provides about the intended state of each instance (pending, running, stopping, etc.) as well as the utilization metrics that Amazon CloudWatch monitors (CPU utilization, network traffic, and disk activity).

Status checks are performed every five minutes and each returns a pass or a fail status. If all checks pass, the overall status of the instance is **OK**. If one or more checks fail, the overall status is **impaired**. There are two types of status checks: system status checks and instance status checks.

System status checks monitor the AWS systems required to use your instance to ensure they are working properly. These checks detect problems with your instance that require AWS involvement to repair. When a system status check fails, you can choose to wait for AWS to fix the issue or you can

resolve it yourself (for example, by stopping and restarting or terminating and replacing an instance). Examples of problems that cause system status checks to fail include:

- Loss of network connectivity
- Loss of system power
- Software issues on the physical host
- Hardware issues on the physical host

Instance status checks monitor the software and network configuration of your individual instance. These checks detect problems that require your involvement to repair. When an instance status check fails, typically you will need to address the problem yourself (for example by rebooting the instance or by making modifications in your operating system). Examples of problems that may cause instance status checks to fail include:

- Failed system status checks
- Misconfigured networking or startup configuration
- Exhausted memory
- Corrupted file system
- Incompatible kernel

Note

Status checks that occur during instance reboot or while a Windows instance store-backed instance is being bundled will report an instance status check failure until the instance becomes available again.

Viewing Status

AWS provides you with several ways to view and work with status checks: You can use the AWS Management Console, interact directly with the API, or use the command line interface.

AWS Management Console

To view status checks

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the Navigation pane, click **Instances**.
3. On the **Instances** page, the **Status Checks** column lists the operational status of each instance.

Name	Instance	AMI ID	Root Device	Type	State	Status Checks	Monitoring
1 FAH r	i-34c4755e	ami-bcd53ed5	ebs	c1.medium	running	2/2 checks passed	detailed
FAH1 P	i-bd7670d7	ami-d9e40db0	ebs	m2.4xlarge	running	2/2 checks passed	detailed
2 FAH b	i-373d3b5d	ami-d9e40db0	ebs	t1.micro	running	1/2 checks passed	detailed
FAH xla	i-bd2764d0	ami-2547a34c	instance-store	c1.xlarge	running	2/2 checks passed	detailed

4. To view an individual instance's status, select the instance, and then click the **Status Checks** tab.

Note

If you have an instance with a failed status check and the instance has been unreachable for over 20 minutes, you can click **Contact Support** to submit a request for assistance. To try and troubleshoot system or instance status check failures yourself, see Troubleshooting Instances with Failed Status Checks (p. 323).

Using the Command Line

Run this command	To do this
`ec2-describe-instance-status`	Return the status of all instances
`ec2-describe-instance-status --filter "instance-status.status=impaired"`	Returns the status of all instances with a instance status of impaired
`ec2-describe-instance-status status -i-15a4417c`	Returns the status of all instances with a single instance with instance ID i-15a4417c

For more information about using the `ec2-describe-instance-status` command, see ec2-describe-instance-status in the *Amazon Elastic Compute Cloud Command Reference Guide.*

Note

If you have an instance with a failed status check, see Troubleshooting Instances with Failed Status Checks (p. 323)

Using the API

You can use the `DescribeInstanceStatus` action to retrieve the status of your instances. For more information, see DescribeInstanceStatus in the *Amazon Elastic Compute Cloud API Reference Guide.*

Reporting Status

You can provide feedback about your instances if you are having problems with an instance whose status is not shown as impaired, or to send AWS additional details about the problems you are experiencing with an impaired instance.

We use reported feedback to identify issues impacting multiple customers, but do not respond to individual account issues reported via this form. Providing feedback will not change the status check results that you currently see for this instance.

If you are in need of technical assistance specific to your account, please post your question to the Developer Forums or contact Premium Support.

AWS Management Console

To report status feedback using the management console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the Navigation pane, click **Instances**.
3. On the **Instances** page, click on the instance on which you want to report status.
4. Click the **Status Checks** tab and then click the Submit Feedback button.
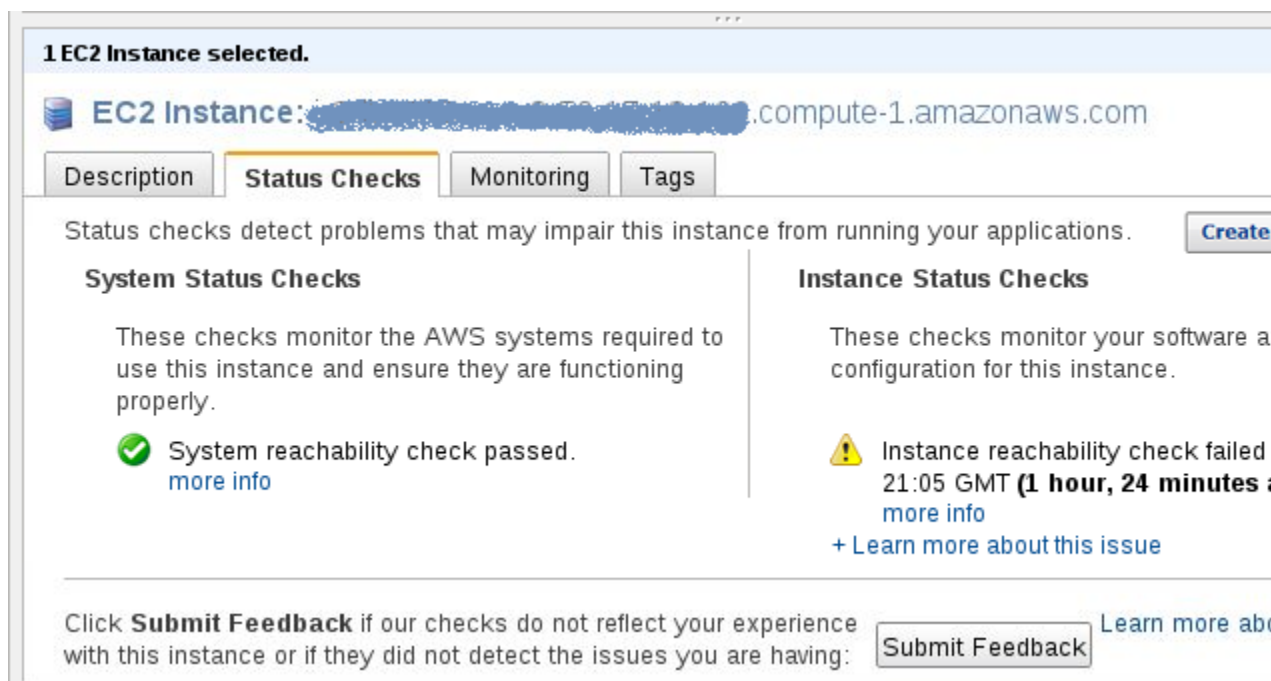
5. Complete the information on the **Report Instance Status** page.

Using the Command Line

Use the `ec2-report-instance-status` command to send status feedback using the command line tools. The command uses the following syntax:

```
ec2-report-instance-status [instance_id ...] [--status ...] [--reason] ..]
```

For more information about using the `ec2-report-instance-status` command, see ec2-report-instance-status in the *Amazon Elastic Compute Cloud Command Reference Guide.*

Using the API

You can use the `ReportInstanceStatus` action to retrieve the status of your instances. For more information, see ReportInstanceStatus in the *Amazon Elastic Compute Cloud API Reference Guide.*

Monitoring Events for Your Instances

Instance status describes specific events that Amazon Web Services (AWS) may schedule for your instances. Events provide information about upcoming activities, such as rebooting or retirement, that are planned for your instances, along with the scheduled start and end time of each event. These scheduled events are not frequent. There are five types of scheduled events:

- **Instance reboot**: AWS may schedule an instance for a reboot for necessary maintenance, such as to apply patch upgrades to an underlying host that *do not* require the host to be rebooted.
- **System reboot**: AWS may schedule an instance for a reboot for necessary maintenance, such as to apply patch upgrades that *do* require the host to be rebooted.
- **Network maintenance**: AWS may schedule network maintenance that includes a scheduled start and end time, during which your instances will not have network connectivity. You will be notified by email if one of your instances is set for network maintenance. The email message indicates when your instance will not have network connectivity.

- **Power maintenance**: AWS may schedule power maintenance that includes a scheduled start and end time, during which your instances may be offline for an extended period and then be rebooted. You will be notified by email if one of your instances is set for power maintenance. The email message indicates when your instance will be rebooted.
- **Instance retirement**: AWS may schedule instances for retirement in cases where there is an unrecoverable issue with the hardware on an underlying host. You will also be notified by email if one of your instances is set to retiring. The email message indicates when your instance will be permanently retired.

> **Important**
>
> No action is required on your part if one of your instances is scheduled for reboot. We recommend that you wait for the reboot to occur within its scheduled maintenance window.
>
> For instances scheduled for network maintenance, power maintenance, or retirement, we recommend that you take the actions detailed later in this section.

Monitoring Events with Instance Status

You can view scheduled events for your instances using the AWS Management Console, the command line interface (CLI), or the API.

AWS Management Console

To view scheduled events for your instances

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the **Navigation** pane, click **Events**. You can see a list of all resources with events associated with them. You can view only instances list of instances that have upcoming events scheduled.
3. Alternatively, you can do the following to view upcoming scheduled events:

    a. In the **Navigation** pane, click the **EC2 Dashboard**.

    b. Under Events, you can see the events associated with your Amazon EC2 instances and volumes.

    c. On the **Events** page, in **Viewing**, select **Instances** to view only instances. You can also filter on specific status types.

Command Line Interface

To view scheduled events for your instances

- Enter the following command:

```
PROMPT> ec2-describe-instance-status
```

Amazon EC2 returns output similar to the following:

```
INSTANCE        i-6d9eaa0c      us-east-1d      running 16      running ok
    active      active
SYSTEMSTATUS    reachability    passed
INSTANCESTATUS  reachability    passed
INSTANCE        i-bf1d7cdc      us-east-1d      running 16      running ok
    active
SYSTEMSTATUS    reachability    passed
```

```
INSTANCESTATUS   reachability     passed
INSTANCE         i-bd1d7cde       us-east-1d       running 16       running ok
    active
SYSTEMSTATUS     reachability     passed
INSTANCESTATUS   reachability     passed
INSTANCE         i-831d7ce0       us-east-1d       running 16       running ok
    retiring          2012-01-02T10:00:00+0000
SYSTEMSTATUS     reachability     passed
INSTANCESTATUS   reachability     passed
EVENT    instance-stop   2012-01-02T10:00:00+0000                  The instance
 is running on degraded hardware
INSTANCE         i-6de0fb0e       us-east-1d       running 16       running ok
    retiring          2012-02-10T08:30:00+0000
SYSTEMSTATUS     reachability     passed
INSTANCESTATUS   reachability     passed
EVENT    instance-retiring   2012-02-10T08:30:00+0000                  The in
stance is running on degraded hardware
INSTANCE         i-5cf7793e       us-east-1c       running 16       running ok
    retiring          2012-01-03T00:00:00+0000
SYSTEMSTATUS     reachability     passed
INSTANCESTATUS   reachability     passed
EVENT    instance-stop   2012-01-03T00:00:00+0000                  The instance
 is running on degraded hardware
```

For more information about using the `ec2-describe-instance-status` command, see
ec2-describe-instance-status in the *Amazon Elastic Compute Cloud Command Reference Guide.*

API

You can use the `DescribeInstanceStatus` action to retrieve the status of your instances. For more
information, see DescribeInstanceStatus in the *Amazon Elastic Compute Cloud API Reference Guide.*

Working with an Instance that Has a Scheduled Event

This section discusses tasks you can perform if your instance has one of the following scheduled events.

- Instance or system reboot
- Network maintenance
- Power maintenance
- Instance retirement

Recommended Tasks for Instances Scheduled for a Reboot

AWS may schedule instances for a reboot to perform maintenance tasks such as patch upgrades to the
software on an underlying host. These scheduled reboots are not frequent. There are two types of reboot
events: system reboot and instance reboot. In either case, your instance will be rebooted. During a system
reboot, the hardware supporting your instance will also be rebooted. During an instance reboot, your
instance will be rebooted but the hardware supporting the instance will not be rebooted.

Important

No action is required on your part if one of your instances is scheduled for reboot. We recommend
that you wait for the reboot to occur automatically within its scheduled maintenance window.

Scheduled reboot events start within their scheduled maintenance window. Once initiated, both system and instance reboots typically complete in a matter of minutes. After a reboot completes, your instance is available to use; it is not necessary to wait until the scheduled end time.

To verify that the reboot has occurred, check your scheduled events and verify that the instance no longer shows a scheduled event. We recommend that you check your instance after it is rebooted to ensure that your application is functioning as you expected.

Optional Alternative Tasks for Instances Scheduled for Reboot

We recommend that you wait for the reboot to occur automatically within its scheduled window. If you choose, you may perform the instance reboot yourself to control the timing of the event.

Instance Reboot

For instance reboot events, you can perform the reboot yourself (be careful to not shut down or terminate your instance) at any time before the scheduled reboot event begins. The reboot can be initiated using the AWS Management Console, a `RebootInstances` API call, or from within the instance (e.g., at the command prompt). After you reboot, any pending maintenance to the underlying host is performed automatically, and you can begin using your instance again after the instance has fully booted.

> **Note**
>
> After you perform the reboot, the scheduled event for the instance reboot is canceled immediately. The event's description is updated in the AWS console to reflect this.

System Reboot

If you choose to perform a system reboot, your course of action will differ depending on whether your instance's root device is an EBS volume or the instance store. You can determine the root device type for an instance using either the DescribeInstances API or the AWS Management Console. In the console, you select an instance and view the root device type listed in the **Description** tab.

Instances Using an EBS Volume as the Root Device Type

If your instance's root device is an EBS volume, you can stop and restart it (be careful not to shut down or terminate your instance). If you choose to do so, some configuration settings will changes.

If you stop and restart your instance, the following changes occur. If you wait for Amazon EC2 to perform the scheduled reboot automatically, these configuration settings remain the same.

* Data in the instance store (i.e., data not stored on any attached EBS volume including the root device) will no longer be available. Before you stop the instance, backup any data you may need.

* The public DNS name and the private IP address of the instance will change.

* If you associated an Elastic IP address with this instance, stopping this instance also disassociates the Elastic IP address from it. Once you start the instance, re-associate it with the Elastic IP address if the address is still required.

> **Note**
>
> When the Elastic IP address is not in use, you will incur charges.

* For instances in Amazon Virtual Private Cloud (Amazon VPC), the Elastic IP address and the private IP address remain unchanged.

Before you stop and restart an instance, perform the following tasks:

1. Retrieve any data from the instance store (i.e., data that is not stored on any attached EBS volumes including the root device) that you will need later. This data will not be available after you stop and restart an instance.

2. Take a snapshot of your existing volume (storage charges will apply).

3. Note the necessary configuration data in case you will need it later, including the DNS public name and the private IP address.

4. Stop and restart your instance. For more information, see Stopping and Starting Instances (p. 312).

5. Re-associate an Elastic IP address if this address is necessary.

6. If other applications or instances rely on the public DNS name or the private IP address of this instance, update the information with the new configuration data.

> **Note**
>
> After you stop and restart the instance, the scheduled event for the system reboot is canceled immediately. The event's description is updated in the AWS console to reflect this.

Instances Using the Instance Store as the Root Device Type

If your instance's root device is the instance store and you cannot wait for AWS to reboot your instance in the scheduled maintenance window, you can opt to launch a replacement instance. After you have launched the replacement instance, you can terminate the original instance. If you choose to take this action, be aware that:

- The public DNS name and the private IP address of your replacement instance will differ from your original instance.

- You must first ensure that you have an AMI with the configuration that you want to use when launching the new instance. For more information, see:
  - ???
  - Creating Instance Store-Backed Linux/UNIX AMIs (p. 39)

Instances Scheduled for Network Maintenance, Power Maintenance, or Retirement

The following section outlines steps you can take to migrate or replace instances that are scheduled for network maintenance, power maintenance, or retirement. If you migrate your instances scheduled for network maintenance, power maintenance or retirement before the maintenance start time, the maintenance event will automatically be cancelled.

Instances Using an EBS Volume as the Root Device Type

If your instance's root device is an EBS volume, you can stop and restart it (be careful not to shut down or terminate the instance). If you choose to do so, some configuration settings will change. The following changes occur:

- Data in the instance store (i.e., data not stored on any attached EBS volume including the root device) will no longer be available. Before you stop the instance, backup any data you may need.

- The public DNS name and the private IP address of the instance will change.

- If you associated an Elastic IP address with this instance, stopping this instance also disassociates the Elastic IP address from it. Once you start the instance, re-associate it with the Elastic IP address if the address is still required.

> **Note**
>
> When the Elastic IP address is not in use, you will incur charges.

- For instances in Amazon Virtual Private Cloud (Amazon VPC), the Elastic IP address and the private IP address remain unchanged.

Before you stop and restart an instance, perform the following tasks:

1. Retrieve any data from the instance store (i.e., data that is not stored on any attached EBS volumes including the root device) that you will need later. This data will not be available after you stop and restart an instance.

2. Take a snapshot of your existing volume (storage charges will apply).

3. Note the necessary configuration data in case you will need it later, including the DNS public name and the private IP address.

4. Stop and restart your instance. For more information, see Stopping and Starting Instances (p. 312).

5. Re-associate an Elastic IP address, if this address is necessary.

6. If other applications or instances rely on the public DNS name or the private IP address of this instance, update the information with the new configuration data.

> **Note**
>
> After you stop and restart the instance, the scheduled event is canceled immediately. The event's description is updated in the AWS console to reflect this.

Instances Using the Instance Store as the Root Device Type

If your instance's root device is the instance store and your instance is scheduled for network maintenance, power maintenance, or retirement, you can opt to launch a replacement instance. After you have launched a replacement instance, you can terminate the original instance. If you choose to take this action, be aware that:

- The public DNS name and the private IP address of your replacement instance will differ from your original instance.

- You must first ensure that you have an AMI with the configuration that you want to use when launching the new instance. For more information, see:

  - ???
  - Creating Instance Store-Backed Linux/UNIX AMIs (p. 39)

Ensuring Idempotency

When you launch an instance through the API or the command line interface tools (CLI), you can optionally provide a client token to ensure the request is idempotent. If timeouts or connection errors occur, you can repeat the request and be sure you haven't launched more instances than you intended. The token must be a unique, case-sensitive string of up to 64 ASCII characters.

If you repeat the request with the same client token, the same response is returned for each repeated request. The only information that might vary in the response is the state of the instance (e.g., pending). The client token is included in the response when you describe the instance.

If you repeat the request with the same client token, but change another request parameter, Amazon EC2 returns an `IdempotentParameterMismatch` error.

The client token is valid for at least 24 hours after the termination of the instance. You should not reuse a client token for another call later on.

You can use the same client token for the same request across different Regions. For example, if you send an idempotent request to launch an instance in the us-east-1 Region, and you use the same exact call (with the same client token) in each of the other Regions, the result is a different running instance in each Region.

The following table shows common response codes and the recommended course of action.

Code	Retry	Comments
200 (OK)	No effect	After you receive a 200, the request has succeeded and any further retries have no effect.
400 (Client Error)	Not recommended	A 400 response typically indicates the request will never succeed (for example when a specified parameter value is not valid). In certain specific cases relating to resources that are transitioning between states, a repeat of the request could possibly succeed (e.g., launching an instance against an Amazon EBS volume that is currently transitioning to the available state).
500 (Server Internal Error)	Recommended	These errors are generally transient. Repeat the request with an appropriate back-off strategy.
503 (Server Unavailable)	Recommended	These errors can occur in times of extreme load. Repeat the request with an appropriate back-off strategy.

Command Line Tools

Use the ec2-run-instances command.

To send an idempotent request to launch instances

- Add the `--client-token` option to your request with a unique, case-sensitive string of up to 64 ASCII characters.

```
PROMPT>  ec2-run-instances ami-b232d0db -k gsg-keypair --client-token
550e8400-e29b-41d4-a716-446655440000
```

API

Use the RunInstances action.

To send an idempotent request to launch instances

- Add the `ClientToken` parameter to your Query request with a unique, case-sensitive string of up to 64 ASCII characters.

```
https://ec2.amazonaws.com/
?Action=RunInstances
&ImageId=ami-3ac33653
...
```

```
&ClientToken=550e8400-e29b-41d4-a716-446655440000
&AUTHPARAMS
```

Instance Metadata

Topics

Amazon EC2 instances can access instance-specific metadata as well as data supplied when launching the instances.

You can use this data to build more generic AMIs that can be modified by configuration files supplied at launch time. For 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.

Metadata is divided into categories. For a list of the categories, see Metadata Categories (p. 308).

Data Retrieval

You can retrieve instance metadata from a running instance using a Query API. The base URI of all requests is `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.

Metadata Retrieval

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).

Examples

The following examples list HTTP GET requests and responses on Linux instances.

Note

On Windows instances, you can install a tool such as cURL or GNU Wget to make these types of metadata requests.

This example gets the available versions of the metadata. These versions do not necessarily correlate with an EC2 API version.

```
GET http://169.254.169.254/
1.0
2007-01-19
2007-03-01
2007-08-29
2007-10-10
2007-12-15
2008-02-01
2008-09-01
2009-04-04
2011-01-01
...
```

This example gets the top-level metadata items. Some of these items are available only for instances in a VPC. For more information about each of these items, see Metadata Categories (p. 308).

```
GET http://169.254.169.254/latest/meta-data/
amiid
ami-launch-index
ami-manifest-path
block-device-mapping/
hostname
instance-action
instance-id
instance-type
kernel-id
local-hostname
local-ipv4
ipv4-associations
mac
network/
placement/
public-hostname
public-ipv4
public-keys/
reservation-id
security-groups
```

This example gets the value of some of the metadata items from the preceding example.

```
GET http://169.254.169.254/latest/meta-data/ami-manifest-path
my-amis/spamd-image.manifest.xml
GET http://169.254.169.254/latest/meta-data/ami-id
ami-2bb65342
GET http://169.254.169.254/latest/meta-data/reservation-id
r-fea54097
GET http://169.254.169.254/latest/meta-data/hostname
ec2-67-202-51-223.compute-1.amazonaws.com
```

This example gets the list of available public keys.

```
GET http://169.254.169.254/latest/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/latest/meta-data/public-keys/0/
openssh-key
```

This example gets public key 0 (in the OpenSSH key format).

```
GET http://169.254.169.254/latest/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/latest/meta-data/product-codes
774F4FF8
```

This example gets an instance's Media Access Control (MAC) address.

```
GET http://169.254.169.254/latest/meta-data/mac
02:29:96:8f:6a:2d
```

This example shows the network information available for a VPC instance.

```
GET http://169.254.169.254/latest/meta-data/network/inter
faces/macs/02:29:96:8f:6a:2d/
local-hostname
local-ipv4s
mac
public-ipv4s
security-group-ids
subnet-id
subnet-ipv4-cidr-block
vpc-id
vpc-ipv4-cidr-block
```

This example gets a VPC instance's subnet ID.

```
GET http://169.254.169.254/latest/meta-data/network/inter
faces/macs/02:29:96:8f:6a:2d/subnet-id
subnet-be9b61d7
```

This example shows the network information available for an EC2 instance (one not running in a VPC).

```
GET http://169.254.169.254/latest/meta-data/network/inter
faces/macs/03:15:28:7g:5b:8a/
local-hostname
local-ipv4s
mac
```

```
public-ipv4s
public-hostname
```

User Data Retrieval

When you launch an instance, you can specify *user data*, which is available for all instances in the reservation to retrieve. You can also add (or modify) user data to Amazon EBS-backed instances when they're stopped. Requests for the user data returns the data as-is (content type `application/x-octetstream`). Many people use user data to configure an instance during launch or even run a configuration script.

> **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/latest/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/latest/user-data
[general]
instances: 4

[instance-0]
s3-bucket: <user_name>

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

You can modify the user data for an Amazon EBS-backed instance while the instance is stopped. For more information, see Modifying Attributes of a Stopped Instance (p. 320).

Use Case: 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
```

The `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.

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

RESERVATION     r-fea54097       598916040194    default
INSTANCE i-3ea74257 ami-2bb65342 pending 0 m1.small 2010-03-19T13:59:03+0000
us-east-1a aki-94c527fd ari-96c527ff monitoring-disabled   ebs
INSTANCE i-31a74258 ami-2bb65342 pending 0 m1.small 2010-03-19T13:59:03+0000
us-east-1a aki-94c527fd ari-96c527ff monitoring-disabled   ebs
INSTANCE i-31a74259 ami-2bb65342 pending 0 m1.small 2010-03-19T13:59:03+0000
us-east-1a aki-94c527fd ari-96c527ff monitoring-disabled   ebs
INSTANCE i-31a7425a ami-2bb65342 pending 0 m1.small 2010-03-19T13:59:03+0000
us-east-1a aki-94c527fd ari-96c527ff monitoring-disabled   ebs
```

After they're launched, all instances have a copy of the user data and the common metadata shown here:

- AMI id: ami-2bb65342
- Reservation ID: r-fea54097
- Public keys: none
- Security group names: default
- Instance type: m1.small

However, each instance has certain unique metadata.

Instance 1

Metadata	Value
instance-id	i-3ea74257
ami-launch-index	0
public-hostname	ec2-67-202-51-223.compute-1.amazonaws.com
public-ipv4	67.202.51.223
local-hostname	ip-10-251-50-35.ec2.internal
local-ipv4	10.251.50.35

Instance 2

Metadata	Value
instance-id	i-31a74258
ami-launch-index	1
public-hostname	ec2-67-202-51-224.compute-1.amazonaws.com
public-ipv4	67.202.51.224
local-hostname	ip-10-251-50-36.ec2.internal
local-ipv4	10.251.50.36

Instance 3

Metadata	Value
instance-id	i-31a74259
ami-launch-index	2
public-hostname	ec2-67-202-51-225.compute-1.amazonaws.com
public-ipv4	67.202.51.225
local-hostname	ip-10-251-50-37.ec2.internal
local-ipv4	10.251.50.37

Instance 4

Metadata	Value
instance-id	i-31a7425a
ami-launch-index	3
public-hostname	ec2-67-202-51-226.compute-1.amazonaws.com
public-ipv4	67.202.51.226
local-hostname	ip-10-251-50-38.ec2.internal
local-ipv4	10.251.50.38

Therefore, an instance can determine its portion of the user-supplied data through the following process.

Metadata Discovery Process

1	Determine the index in the launch group.
	```GET http://169.254.169.254/latest/meta-data/ami-launch-index``` ```1```
2	Retrieve the user data.
	```GET http://169.254.169.254/latest/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]```

Metadata Categories

The data available to instances is categorized into metadata and user-supplied data. The following table lists the categories of metadata.

Data	Description	Version Introduced
`ami-id`	The AMI ID used to launch the instance.	1.0
`ami-launch-index`	The index of this instance in the reservation.	1.0
`ami-manifest-path`	The manifest path of the AMI with which the instance was launched.	1.0
`ancestor-ami-ids`	The AMI IDs of any instances that were rebundled to create this AMI.	2007-10-10
`block-device-mapping/`	Defines native device names to use when exposing virtual devices.	2007-12-15
`iam/info`	Returns information about the last time the instance profile was updated, including the instance's LastUpdated date, InstanceProfileArn, and InstanceProfileId.	2012-06-01
`iam/security-credentials/`	Returns the name of the IAM role associated with the instance.	2012-06-01
`iam/security-credentials /role-name`	Where *role-name* is the name of the IAM role associated with the instance. Returns the temporary security credentials (AccessKeyId, SecretAccessKey, SessionToken, and Expiration) associated with the IAM role.	2012-06-01
`instance-action`	Notifies the instance that it should reboot in preparation for bundling. Possible values: `none` | `shutdown`, `bundle-pending`.	2008-09-01
`instance-id`	The ID of this instance.	1.0
`instance-type`	The type of instance. For more information, see Instance Families and Types (p. 85).	2007-08-29
`kernel-id`	The ID of the kernel launched with this instance, if applicable.	2008-02-01
`local-hostname`	The local hostname of the instance.	2007-01-19
`local-ipv4`	The private IP address of the instance.	1.0
`mac`	The instance's MAC address.	2011-01-01
`network/interfaces/macs/ MAC/local-hostname`	The instance's local host name.	2011-01-01
`network/interfaces/macs/ MAC/local-ipv4s`	The private IP addresses associated with the instance.	2011-01-01
`network/interfaces/macs/ MAC/mac`	The instance's Media Access Control (MAC) address.	2011-01-01

Data	Description	Version Introduced
`network/interfaces/macs/` `MAC/public-ipv4s`	The elastic IP addresses associated with the instance.	2011-01-01
`network/interfaces/macs/` `MAC/security-group-ids`	The IDs of the security groups the instance belongs to. Returned for VPC instances only.	2011-01-01
`network/interfaces/macs/` `MAC/subnet-id`	The ID of the Amazon VPC subnet the instance is in. Returned for VPC instances only.	2011-01-01
`network/interfaces/macs/` `MAC/subnet-ipv4-cidr-block`	The CIDR block of the Amazon VPC subnet the instance is in. Returned for VPC instances only.	2011-01-01
`network/interfaces/macs/` `MAC/vpc-id`	The ID of the VPC the instance is in. Returned for VPC instances only.	2011-01-01
`network/interfaces/macs/` `MAC/vpc-ipv4-cidr-block`	The CIDR block of the VPC the instance is in. Returned for VPC instances only.	2011-01-01
`placement/availability-z one`	The Availability Zone in which the instance launched.	2008-02-01
`product-codes`	Product code associated with the instance, if any.	2007-03-01
`public-hostname`	The public hostname of the instance. Not returned for instances in a VPC (they have only private IP addresses).	2007-01-19
`public-ipv4`	The public IP address. If an elastic IP address is associated with the instance, the value returned is the elastic IP address.	2007-01-19
`public-keys`	Public keys. Only available if supplied at instance launch time.	1.0
`ramdisk-id`	The ID of the RAM disk launched with this instance, if applicable.	2007-10-10
`reservation-id`	ID of the reservation.	1.0
`security-groups`	The names of the security groups the instance is launched in. Only available if supplied at instance launch time. For instances in a VPC, returns the IDs of the security groups the instance is in. If you change the instance's group membership, the metadata updates accordingly.	1.0

You can provide user data when you launch an instance, or when the instance is in a stopped state (for EBS-backed instances). 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 in base64 and is decoded before being presented to the instance.

Stopping Instances

Topics

You can stop instances that use Amazon EBS volumes as their root devices. When you stop an instance, it's shut down, so any data stored in the RAM of the host computer is not preserved. You're not billed for hourly usage or data transfer, but you're billed for the storage for any Amazon EBS volumes. You can restart the instance at any time with a start request. Each time you transition an instance from stopped to started, we charge a full instance hour, even if transitions happen multiple times within a single hour.

Instance Stop

When you stop a running instance, the following things happen:

- The instance performs a normal shutdown and stops running (the status goes to *stopping* and then *stopped*)
- The instance retains its instance ID
- Any Amazon EBS volumes remain attached and the data persists
- Any instance store volumes don't persist
- It's likely that the instance won't retain its public and private IP addresses. The exception is instances launched in Amazon VPC; they keep the same IP address when stopped and restarted.
- Any Elastic IP address mapped to the instance is unmapped. For Amazon VPC instances, the Elastic IP address remains associated with the stopped instance. When you restart the instance, the Elastic IP address is active. While the Elastic IP address is associated with the stopped instance, you're charged for the Elastic IP address.
- If you've registered the instance with a load balancer, it's likely that the load balancer won't be able to route traffic to your instance after you've stopped and restarted it. You must de-register the instance from the load balancer after stopping the instance, and then re-register after starting the instance. For more information, see De-Registering and Registering Amazon EC2 Instances section in the *Elastic Load Balancer Developer Guide*.
- When a Windows instance is stopped and restarted, the drive letters mapped to the attached EBS volumes are likely to change. For information on changing this default behavior, see Using Ec2Config Service in the *Microsoft Windows Guide*.

Let's go into more detail for some of the areas covered in the preceding list.

Network

When you stop an instance, it'll probably lose its public and private IP addresses and be given new ones when you restart the instance. Exception: instances you launch with Amazon VPC retain the same private IP address when stopped and then started; it's therefore possible to have an Amazon VPC subnet with no running instances (they're all stopped), and also no remaining IP addresses available.

When you restart a Windows instance, by default, Ec2ConfigService changes the instance host name to match the new IP address and initiates a reboot. For more information about this service, see ???.

If you were using an Elastic IP address with the instance, the address is automatically unmapped when the instance stops. While the instance is stopped, you're charged for the address being unmapped (unless you remap it to another instance). When you restart the instance, you need to manually remap the Elastic IP address to the instance; it doesn't happen automatically.

For Amazon VPC instances, the Elastic IP address remains associated with the stopped instance. When you restart the instance, the Elastic IP address is active. While the Elastic IP address is associated with the stopped instance, you will be charged for the Elastic IP address. For more information about the charges for Elastic IP addresses, go to the Amazon EC2 product page.

Instance Attributes

Each Amazon EBS-backed instance has an attribute called `InstanceInitiatedShutdownBehavior` that controls whether the instance stops or terminates when you initiate a shutdown from within the instance itself. The default is stop. You can modify the attribute while the instance is running or stopped. For more information, see Changing the Instance Initiated Shutdown Behavior (p. 319).

You can modify these instance attributes only while the instance is stopped:

- Kernel
- RAM disk
- Instance type
- User data

For more information about modifying instance attributes while the instance is stopped, see Modifying Attributes of a Stopped Instance (p. 320).

Storage

While the instance is stopped, you can treat the root volume like any other volume, and modify it (for example, repair file system problems or update software). You just detach the volume from the stopped instance, attach it to a running instance, make your changes, detach it from the running instance, and then reattach it to the stopped instance. Make sure you reattach it to the correct storage device (whichever device name is specified as the root device in the instance's block device mapping).

While the instance is stopped, you can't change anything about the instance's block device mapping except the `DeleteOnTermination` flag for an attached volume (you can also change the flag while the instance is running). For more information, see Changing the Root Volume to Persist (p. 122).

Instance Termination

When an instance terminates, any instance store associated with that instance is deleted.

By default, any volumes that were created when the instance launched (the root device and any others specified in the block device mapping) are automatically deleted when the instance terminates (in other words, their `DeleteOnTermination` flags are set to `true` by default). However, any volumes that you attached *after* the instance was running are not (their `DeleteOnTermination` flags are set to `false` by default). If you detach and then reattach a volume that was created at instance launch, it's treated like a new volume that you attached after the launch, and its `DeleteOnTermination` flag is set to `false`.

You can see the value for the `DeleteOnTermination` flag for each of the attached volumes by looking at the instance's block device mapping (for more information, see Viewing the EBS Volumes in an Instance Block Device Mapping (p. 497)). You currently can't change the flag's value after the instance is launched; however, you can launch the instance with your desired value for the flag. In the launch request, you just specify a block device mapping with the value. For an example of how to change the flag at launch time, see Changing the Root Volume to Persist (p. 122).

What happens to the volume when you terminate the instance it's attached to? Assuming the volume's `DeleteOnTermination` flag is `false`, the volume persists in its current state. You could take a snapshot of it, and you could attach it to any other instance you have.

Note

When you launch an instance of a particular AMI and then terminate the instance, any remaining volumes are not automatically related to any future instances of that AMI. In other words, a new instance that you launch of that same AMI doesn't attempt to attach to the remaining volume. However, you can manually attach the remaining volume to the new instance if you want.

For more information on persisting the data after instance termination, see Data Persistence after Instance Termination (p. 437).

Starting with the 2009-10-31 API version, you can prevent an instance from being terminated. This feature is available for both Amazon EC2 instance store-backed and Amazon EBS-backed instances. Each instance has a `DisableApiTermination` attribute that defaults to `false` (i.e., the instance can be terminated). You can modify this instance attribute while the instance is running or stopped (in the case of Amazon EBS-backed instances).

Stopping and Starting Instances

This topic describes how to stop and restart Amazon EC2 instances that use Amazon EBS volumes as their root devices.

Note

If you try to stop an instance that uses an instance store as its root device, you receive an error.

When an instance is stopped, it is shut down, and any data stored in RAM is not preserved. You're not billed for hourly usage or data transfer, but you're billed for any Amazon EBS volume storage. You can start the instance at any time with a start request. Each time you transition an instance from stopped to started, we charge a full instance hour, even if transitions happen multiple times within a single hour.

For information about the differences between stopping an instance and terminating an instance, see Stopping Instances (p. 310).

AWS Management Console

To stop and start an instance

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Click **Instances** in the **Navigation** pane.
   The console displays a list of running instances.
3. Select an instance, select **Instance Actions**, and click **Stop Instance**.
   A confirmation dialog box appears.
4. Click the **Yes, Stop Instance**.
   The instance is stopped and saved as a snapshot.
5. To restart the stopped instance, select **Stopped Instances** from the **Viewing** list box.

The newly stopped instance appears in the list.

6. Select the instance, select **Instance Actions**, and click **Start Instance**.
   The instance begins restarting.

Command Line Tools

To stop and start an instance

1. Use the ec2-stop-instances command to stop an instance.

   ```
   PROMPT>  ec2-stop-instances i-10a64379
   ```

   Amazon EC2 returns output similar to the following example.

   ```
   IMAGE i-10a64379 running stopping
   ```

2. Use the ec2-start-instances command to restart the instance.

   ```
   PROMPT>  ec2-start-instances i-10a64379
   ```

   Amazon EC2 returns output similar to the following example.

   ```
   IMAGE i-10a64379 stopped pending
   ```

API

To stop and start an instance

1. Construct the following Query request to stop an instance.

   ```
   https://ec2.amazonaws.com/
   ?Action=StopInstances
   &InstanceId.1=i-10a64379
   &AUTHPARAMS
   ```

   The following is an example response.

   ```
   <StopInstancesResponse xmlns="http://ec2.amazonaws.com/doc/2012-07-20/">
     <requestId>59dbff89-35bd-4eac-99ed-be587EXAMPLE</requestId>
     <instancesSet>
       <item>
         <instanceId>exampleinstanceid</instanceId>
         <currentState>
           <code>64</code>
           <name>stopping</name>
         </currentState>
         <previousState>
           <code>16</code>
           <name>running</name>
   ```

```
            </previousState>
        </instancesSet>
</StopInstancesResponse>
```

2.   Construct the following Query request to restart the instance.

```
https://ec2.amazonaws.com/
?Action=StartInstances
&InstanceId.1=i-10a64379
&AUTHPARAMS
```

The following is an example response.

```
<StartInstancesResponse xmlns="http://ec2.amazonaws.com/doc/2012-07-20/">
    <requestId>59dbff89-35bd-4eac-99ed-be587EXAMPLE</requestId>
    <instancesSet>
        <item>
            <instanceId>i-10a64379</instanceId>
            <currentState>
                <code>0</code>
                <name>pending</name>
            </currentState>
            <previousState>
                <code>80</code>
                <name>stopped</name>
            </previousState>
        </item>
    </instancesSet>
</StartInstancesResponse>
```

Terminating Instances

This topic describes how to terminate an Amazon EC2 instance.

If the instance you launched is not in the free usage tier, as soon as your instance starts to boot, you're billed for each hour or partial hour that you keep the instance running (even if the instance is idle). When you've decided that you no longer need an instance, you can terminate it. As soon as the status of an instance changes to `shutting down` or `terminated`, you stop incurring charges for that instance.

For information about the differences between stopping an instance and terminating an instance, see Stopping Instances (p. 310). For information about preventing an instance from being terminated, see Enabling Termination Protection for an Instance (p. 315).

You cannot restart a terminated instance. However, you can launch additional instances of the same AMI.

AWS Management Console

To terminate an instance

1.   Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2.   In the Navigation pane, click **Instances**.

3.   Right-click the instance, and then click **Terminate**.

4.   Click **Yes, Terminate** when prompted for confirmation.
     Amazon EC2 begins terminating the instance.

Command Line Tools

To terminate an instance, use the ec2-terminate-instances command. This example command terminates
the instance with the ID i-10a64379.

```
ec2-terminate instances i-10a64379
```

The following is an example of the output for this command.

```
INSTANCE i-10a64379 running shutting-down
```

API

To terminate an instance, use the TerminateInstances command. This example Query terminates the
instance with the ID i-10a64379.

```
https://ec2.amazonaws.com/?Action=TerminateInstances
&InstanceId.1=i-10a64379
&AUTHPARAMS
```

The following is an example response.

```
<TerminateInstancesResponse xmlns="http://ec2.amazonaws.com/doc/2012-07-20/">

  <requestId>59dbff89-35bd-4eac-99ed-be587EXAMPLE</requestId>
  <instancesSet>
    <item>
      <instanceId>i-10a64379</instanceId>
      <currentState>
        <code>32</code>
        <name>shutting-down</name>
      </currentState>
      <previousState>
        <code>16</code>
        <name>running</name>
      </previousState>
    </item>
  </instancesSet>
</TerminateInstancesResponse>
```

Enabling Termination Protection for an Instance

Note

The following information applies to instances of Amazon EBS-backed AMIs or Amazon EC2
instance store-backed AMIs.

By default, you can terminate any instances you launch. If you want to prevent accidental termination of the instance, you can enable *termination protection* for the instance. The instance's `DisableApiTermination` attribute controls whether the instance is protected. A value of `true` means the instance is protected and can't be terminated; a value of `false` means it's not protected and can be terminated. You can specify whether an instance is protected either at launch time or after the instance is running. You can modify this attribute while the instance is running or stopped (for Amazon EBS-backed instances).

AWS Management Console

By default, termination protection is disabled for an instance at launch time.

To enable termination protection for an instance at launch time

- In the **Request Instances Wizard**, select the check box for **Termination Protection**.
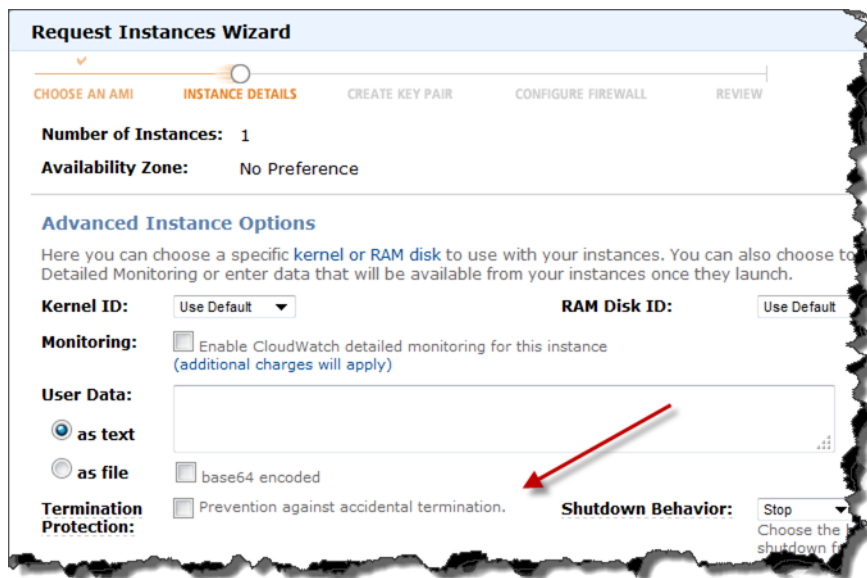
To enable termination protection for an instance after it's launched

1. In your list of instances, right-click the instance and select **Change Termination Protection**.
    A confirmation dialog box appears with the current status of termination protection.

2. Click **Yes, Enable**.

    Now the instance cannot be terminated until you disable termination protection.

You can use the same procedure to disable termination protection for the instance. You can enable and disable this attribute of an instance as often as you want.

Command Line Tools

By default, termination protection is disabled for an instance at launch time.

To enable termination protection for an instance at launch time

- Add `--disable-api-termination` to the `ec2-run-instances` command.

```
PROMPT>  ec2-run-instances ami_id --disable-api-termination ...
```

To enable termination protection for an instance after it's launched

1. View the current value for the attribute with the following command.

    ```
    PROMPT>  ec2-describe-instance-attribute instance_id --disable-api-termination
    ```

    The following is a sample response.

    ```
    disableApiTermination   i-87ad5eec      false
    ```

2. Enable termination protection with the following command.

    ```
    PROMPT>  ec2-modify-instance-attribute instance_id --disable-api-termination
     true
    ```

    The following is a sample response.

    ```
    disableApiTermination   i-87ad5eec      true
    ```

To disable termination protection for an instance

*   Disable termination protection for the instance with the following command.

    ```
    PROMPT>  ec2-modify-instance-attribute instance_id --disable-api-termination
     false
    ```

    The following is a sample response.

    ```
    disableApiTermination   i-87ad5eec      false
    ```

You can enable and disable this attribute of an instance as often as you want.

API

By default, termination protection is disabled for an instance at launch time.

To enable termination protection for an instance at launch time

*   Issue a Query request for `RunInstances` similar to the following example and include `DisableApiTermination=true`.

```
https://ec2.amazonaws.com/?Action=RunInstances
&ImageId=ami-60a54009
&MaxCount=3
```

```
&MinCount=1
&DisableApiTermination=true
&Placement.AvailabilityZone=us-east-1b
&Monitoring.Enabled=true
&AUTHPARAMS
```

For information about the auth parameters, go to Common Query Parameters in the *Amazon Elastic Compute Cloud API Reference*.

To enable termination protection for an instance after it's launched

1. Issue the following Query request to get the current value of the `disableApiTermination` attribute.

```
https://ec2.amazonaws.com/
?Action=DescribeInstanceAttribute
&InstanceId=i-87ad5eec
&Attribute=disableApiTermination
&AUTHPARAMS
```

The following is an example response.

```
<DescribeInstanceAttributeResponse xmlns="http://ec2.amazonaws.com/doc/2012-
07-20/">
    <requestId>59dbff89-35bd-4eac-99ed-be587EXAMPLE</requestId>
    <instanceId>i-87ad5eec</instanceId>
    <disableApiTermination>
        <value>false</value>
    </disableApiTermination>
</DescribeInstanceAttributeResponse>
```

2. Issue the following Query request to modify the `disableApiTermination` attribute.

```
https://ec2.amazonaws.com/
?Action=ModifyInstanceAttribute
&InstanceId=i-87ad5eec
&DisableApiTermination.Value=true
&AUTHPARAMS
```

The following is an example response.

```
<ModifyInstanceAttributeResponse xmlns="http://ec2.amazonaws.com/doc/2012-
07-20/">
    <requestId>59dbff89-35bd-4eac-99ed-be587EXAMPLE</requestId>
    <return>true</return>
</ModifyInstanceAttributeResponse>
```

To disable termination protection for an instance

- Issue the following Query request to modify the `disableApiTermination` attribute.

```
https://ec2.amazonaws.com/
?Action=ModifyInstanceAttribute
&InstanceId=i-87ad5eec
&DisableApiTermination.Value=false
&AUTHPARAMS
```

The following is an example response.

```
<ModifyInstanceAttributeResponse xmlns="http://ec2.amazonaws.com/doc/2012-
07-20/">
    <return>true</return>
</ModifyInstanceAttributeResponse>
```

You can enable and disable this attribute of an instance as often as you want.

Changing the Instance Initiated Shutdown Behavior

By default, when you initiate a shutdown from within an EBS-backed instance, the instance stops. You can change this so the instance terminates instead. You just modify the `InstanceInitiatedShutdownBehavior` attribute for the instance. You can do this while the instance is either running or stopped.

Command Line tools

To change the InstanceInitiatedShutdownBehavior attribute

1. View the current value for the flag with the following command.

    ```
    PROMPT>  ec2-describe-instance-attribute instance_id --instance-initiated-
    shutdown-behavior
    ```

    Sample response:

    ```
    instanceInitiatedShutdownBehavior       i-87ad5eec       stop
    ```

2. Change the flag's value to `terminate` with the following command.

    ```
    PROMPT>  ec2-modify-instance-attribute instance_id --instance-initiated-
    shutdown-behavior terminate
    ```

    Sample response:

    ```
    instanceInitiatedShutdownBehavior i-87ad5eec terminate
    ```

You can toggle the attribute between `stop` and `terminate` as often as you want.

API

To change the InstanceInitiatedShutdownBehavior attribute

1. Issue the following Query request to get the current value of the attribute.

```
https://ec2.amazonaws.com/
?Action=DescribeInstanceAttribute
&InstanceId=i-87ad5eec
&Attribute=instanceInitiatedShutdownBehavior
&AUTHPARAMS
```

For information about the auth parameters, go to Common Query Parameters in the *Amazon Elastic Compute Cloud API Reference*.

The following is an example response.

```
<DescribeInstanceAttributeResponse xmlns="http://ec2.amazonaws.com/doc/2012-
07-20/">
    <requestId>59dbff89-35bd-4eac-99ed-be587EXAMPLE</requestId>
    <instanceId>i-5d7e8e36</instanceId>
    <instanceInitiatedShutdownBehavior>
        <value>stop</value>
    </instanceInitiatedShutdownBehavior>
</DescribeInstanceAttributeResponse>
```

2. Issue the following Query request to change the attribute's value.

```
https://ec2.amazonaws.com/
?Action=ModifyInstanceAttribute
&InstanceId=i-87ad5eec
&Attribute=instanceInitiatedShutdownBehavior
&Value=terminate
&AUTHPARAMS
```

The following is an example response.

```
<ModifyInstanceAttributeResponse xmlns="http://ec2.amazonaws.com/doc/2012-
07-20/">
    <requestId>59dbff89-35bd-4eac-99ed-be587EXAMPLE</requestId>
    <return>true</return>
</ModifyInstanceAttributeResponse>
```

You can toggle the attribute between `stop` and `terminate` as often as you want.

Modifying Attributes of a Stopped Instance

Instances have a set of attributes, and you can modify the following attributes only when the instance is stopped:

- Kernel
- RAM disk

- Instance type
- User data

If you try to modify one of these attributes while the instance is running, Amazon EC2 returns the `IncorrectInstanceState` error.

For a list of all the available instance attributes you can change (not just those that require the instance to be stopped), go to ec2-modify-instance-attribute in the *Amazon Elastic Compute Cloud Command Line Reference*.

Note

Although you can use the AWS Management Console to stop and start instances, and to change the instance type or user data, you can't use it to modify the kernel or RAM disk instance attributes. That functionality is available only through the command line tools or API.

For information about kernels and RAM disks, see Kernels and RAM Disk FAQ (p. 555).

AWS Management Console

To change the instance type for a stopped instance

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the **Navigation** pane, click **Instances**.
3. In the **My Instances** pane, right-click a stopped instance, and then click **Change Instance Type**.
4. In the **Change Instance Type** dialog box, in the **Instance Type** drop-down list, select the type of instance you want, and then click **Yes, Change**.

To change the user data for a stopped instance

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the **Navigation** pane, click **Instances**.
3. In the **My Instances** pane, right-click a stopped instance, and then click **View/Change User Data**.
4. In the **View/Change User Data** dialog box, update the user data, and then click **Yes, Change**.

Command Line Tools

You can modify only a single attribute with each command.

To change an attribute of a stopped instance

1. Choose one of the following commands to get the current value of the attribute of interest.

   ```
   PROMPT>  ec2-describe-instance-attribute instance_id --kernel
   ```

   ```
   PROMPT>  ec2-describe-instance-attribute instance_id --ramdisk
   ```

   ```
   PROMPT>  ec2-describe-instance-attribute instance_id --instance-type
   ```

   ```
   PROMPT>  ec2-describe-instance-attribute instance_id --user-data
   ```

The following is a sample response from the first command in the preceding list.

```
kernel  i-87ad5eec      aki-94c527fd
```

2. Choose one of the following commands to modify the value of the attribute of interest.

```
PROMPT>  ec2-modify-instance-attribute instance_id --kernel kernel_id
```

```
PROMPT>  ec2-modify-instance-attribute instance_id --ramdisk ramdisk_id
```

```
PROMPT>  ec2-modify-instance-attribute instance_id --instance-type in
stance_type
```

```
PROMPT>  ec2-modify-instance-attribute instance_id --user-data user_data
```

When you restart the instance, the new attribute value takes effect.

API

You can modify only a single attribute in each call.

To change an attribute of a stopped instance

1. Issue the following Query request to first get the current value of the attribute of interest. The example gets the current value for the instance type.

```
https://ec2.amazonaws.com/
?Action=DescribeInstanceAttribute
&InstanceId=i-87ad5eec
&Attribute=instanceType
&AUTHPARAMS
```

For information about the auth parameters, go to Common Query Parameters in the *Amazon Elastic Compute Cloud API Reference*.

The following is an example response.

```
<DescribeInstanceAttributeResponse xmlns="http://ec2.amazonaws.com/doc/2012-
07-20/">
    <requestId>59dbff89-35bd-4eac-99ed-be587EXAMPLE</requestId>
  <instanceId>i-5d7e8e36</instanceId>
  <instanceType>
     <value>m1.small</value>
  </instanceType>
</DescribeInstanceAttributeResponse>
```

2. Issue the following Query request to modify the attribute of interest. The example changes the instance type to c1.medium.

```
https://ec2.amazonaws.com/
?Action=ModifyInstanceAttribute
&InstanceId=i-87ad5eec
&Attribute=instanceType
&Value=c1.medium
&AUTHPARAMS
```

The following is an example response.

```
<ModifyInstanceAttributeResponse xmlns="http://ec2.amazonaws.com/doc/2012-
07-20/">
    <return>true</return>
</ModifyInstanceAttributeResponse>
```

Related Topics

- Kernels and RAM Disk FAQ (p. 555)
- Instance Types and Architectures FAQ (p. 544)
- User Data Retrieval (p. 305)

Troubleshooting Instances

Topics

- Troubleshooting Instances with Failed Status Checks (p. 323)
- What to Do If an Instance Immediately Terminates (p. 346)
- Getting Console Output and Rebooting Instances (p. 347)

This section provides information to help you troubleshoot problems with your instance.

Troubleshooting Instances with Failed Status Checks

Initial Steps You Can Take

Investigating Impaired Instances Using the AWS Management Console

If your instance fails a status check, first determine whether your applications are exhibiting any problems. If you verify that the instance is not running your applications as expected, follow these steps:

1. Log into the AWS Management Console.
2. In the **Amazon EC2** console, under My Instances, select your instance.
3. Click the **Status Checks** tab in the lower pane to see the individual results for all **System Status Checks** and **Instance Status Checks**.

If a system status check has failed, you can try one of the following options:

- For an instance using an EBS-backed AMI, stop and re-start the instance.

- For an instance using an instance-store backed AMI, terminate the instance and launch a replacement.
- Wait for our systems to resolve the issue.
- If these steps do not resolve your issue, you can seek assistance by posting your issue to the Developer Forums or contacting Premium Support.

If an instance status check has failed, follow these steps:

1. Right-click your instance, and then click **Reboot**. It may take a few minutes for your system to restart.
2. Verify that the problem still exists; in some cases, rebooting may resolve the problem.
3. Wait until the instance shows a `running` state.
4. Right-click the instance and then click **Get System Log**. You can use this information to help identify the problem. Be sure you have rebooted recently to clear unnecessary information from the log.
5. Review the log that appears on the screen.
6. Use the list of known system log error statements below to troubleshoot your issue.
7. If these steps do not resolve your issue, you can seek assistance by posting your issue to the Developer Forums or contact Premium Support.

If your experience differs from the our check results, or if you are having an issue with your instance that our checks did not detect, click **Report an Issue** at the bottom of the **Status Checks** tab to help us improve our detection tests.

Troubleshooting Instance Status Checks for Linux-Based Instances

For Linux-based instances that have failed an instance status check, such as the instance reachability check, first be sure you have followed the steps discussed earlier to retrieve the system log. The following list contains some common system log errors and suggested actions you can take to resolve the issue for each error .

Memory Errors

- Out of memory error (OOM): "Out of memory: kill process" (p. 325)
- Memory management update failed: "ERROR: mmu_update failed" (p. 326)

Device Errors

- "I/O error, … , sector " (Block device failure) (p. 326)
- "IO ERROR: neither local nor remote disk" (Broken distributed block device) (p. 328)

Kernel Errors

- "request_module: runaway loop modprobe" (Looping legacy kernel modprobe that applies to older Linux versions, e.g. 2.6.16-xenU) (p. 329)
- "FATAL: kernel too old" and "fsck. …: No such file or directory while trying to open /dev" (Kernel and AMI mismatch) (p. 329)
- "FATAL: Could not load /lib/modules" or "BusyBox" (Missing kernel modules) (p. 330)
- "ERROR Invalid kernel" (EC2 incompatible kernel) (p. 332)

File System Errors

- "request_module: runaway loop modprobe " (Looping legacy kernel modprobe that applies to older Linux versions, e.g. 2.6.16-xenU) (p. 333)
- "fsck. … : No such file or directory while trying to open /dev" (Filesystem not found) (p. 333)
- "General error mounting filesystems" (Failed Mount) (p. 335)
- "Kernel panic - not syncing: VFS: Unable to mount root fs on unknown-block" (Root filesystem mismatch) (p. 337)
- "Error: Unable to determine major/minor number of root device…" or "[ramfs /]#" (Root filesystem/device mismatch) (p. 338)
- "XENBUS: Device with no driver: device/vif/0"" (p. 339)
- Filesystem check required: "… days without being checked, check forced." (p. 340)
- "fsck died with exit status…" (Missing device). (p. 340)

Operating System Errors

- GRUB prompt: "grubdom>") (p. 341)
- "Bringing up interface eth0: Device eth0 has different MAC address than expected, ignoring. [FAILED]" (Hardcoded MAC address) (p. 343)
- "Unable to load SELinux Policy. Machine is in enforcing mode. Halting now. … Kernel panic - not syncing: Attempted to kill init!" (SELinux misconfiguration) (p. 344)
- " XENBUS: Timeout connecting to devices!" ((Xenbus timeout)) (p. 345)

Memory Errors

Out of memory error (OOM): "Out of memory: kill process"

An out of memory error is indicated by a system log entry similar to the one shown below:

```
[115879.769795] Out of memory: kill process 20273 (httpd) score 1285879
or a child
[115879.769795] Killed process 1917 (php-cgi) vsz:467184kB, anon-
rss:101196kB, file-rss:204kB
```

Potential Cause

Exhausted memory

Suggested Actions

For this instance type	Do this
EBS-backed	Do one of the following: • Stop the instance, and modify the instance to use a different instance type, and start the instance again. For example, you may want to use a larger instance type or a high-memory instance type. • Reboot the instance to return it to a non-impaired status. The problem will probably occur again unless you change the instance type.

For this instance type	Do this
Instance-store backed	Do one of the following: • Relaunch the instance using a different instance type. For example, you may want to use a larger instance type or a high-memory instance type. • Reboot the instance to return it to an unimpaired status. The problem will probably occur again unless you change the instance type.

Memory management update failed: "ERROR: mmu_update failed"

Memory management update failures are indicated by a system log entry similar to the following:

```
...
Press `ESC' to enter the menu... 0   [H[J  Booting 'Amazon Linux 2011.09
(2.6.35.14-95.38.amzn1.i686)'


root (hd0)

 Filesystem type is ext2fs, using whole disk

kernel /boot/vmlinuz-2.6.35.14-95.38.amzn1.i686 root=LABEL=/ console=hvc0 LANG=

en_US.UTF-8 KEYTABLE=us

initrd /boot/initramfs-2.6.35.14-95.38.amzn1.i686.img

ERROR: mmu_update failed with rc=-22
```

Potential Cause

Issue with Amazon Linux.

Suggested Action

Seek assistance by posting your issue to the Developer Forums or contacting Premium Support.

Device Errors

"I/O error, … , sector " (Block device failure)

An input/output error is indicated by a system log entry similar to the following example:

```
[9943662.053217] end_request: I/O error, dev sde, sector 52428288
[9943664.191262] end_request: I/O error, dev sde, sector 52428168
[9943664.191285] Buffer I/O error on device md0, logical block 209713024
[9943664.191297] Buffer I/O error on device md0, logical block 209713025
[9943664.191304] Buffer I/O error on device md0, logical block 209713026
[9943664.191310] Buffer I/O error on device md0, logical block 209713027
[9943664.191317] Buffer I/O error on device md0, logical block 209713028
```

```
[9943664.191324] Buffer I/O error on device md0, logical block 209713029
[9943664.191332] Buffer I/O error on device md0, logical block 209713030
[9943664.191339] Buffer I/O error on device md0, logical block 209713031
[9943664.191581] end_request: I/O error, dev sde, sector 52428280
[9943664.191590] Buffer I/O error on device md0, logical block 209713136
[9943664.191597] Buffer I/O error on device md0, logical block 209713137
[9943664.191767] end_request: I/O error, dev sde, sector 52428288
[9943664.191970] end_request: I/O error, dev sde, sector 52428288
[9943664.192143] end_request: I/O error, dev sde, sector 52428288
[9943664.192949] end_request: I/O error, dev sde, sector 52428288
[9943664.193112] end_request: I/O error, dev sde, sector 52428288
[9943664.193266] end_request: I/O error, dev sde, sector 52428288
...
```

Potential Causes

Instance type	Potential cause
EBS-backed	Failed EBS volume
Instance-store backed	Failed physical drive

Suggested Actions

For this instance type	Do this
EBS-backed	1. Stop the instance.
	2. Detach the volume.
	3. Attempt to recover the volume.
	4. Re-attach the volume to the instance.
	5. Detach the volume.
	Option 2: Terminate the instance and relaunch the instance.
	Note
	Data will not be recovered.
	Tip
	It's good practice to snapshot your EBS-backed volumes often. This dramatically decreases the risk of data loss as a result of failure.

For this instance type	Do this
Instance-store backed	Terminate the instance and relaunch the instance. **Note** Data will not be recovered. Recover from backups **Tip** It's a good practice to use either Amazon S3 or Amazon EBS volumes for backups. Non-EBS volumes are directly tied to single host and single disk failures.

"IO ERROR: neither local nor remote disk" (Broken distributed block device)

An input/output error on the device is indicated by a system log entry similar to the following example:

```
...
block drbd1: Local IO failed in request_timer_fn. Detaching...

Aborting journal on device drbd1-8.

block drbd1: IO ERROR: neither local nor remote disk

Buffer I/O error on device drbd1, logical block 557056

lost page write due to I/O error on drbd1

JBD2: I/O error detected when updating journal superblock for drbd1-8.
```

Potential Causes

Instance type	Potential cause
EBS-backed	Failed EBS volume
Instance-store backed	Failed physical drive

Suggested Action

Terminate the instance and launch another instance.

For an EBS-backed instance you can recover data from a recent snapshot by creating an image from it. Any data added after the snapshot will not be recovered.

Kernel Errors

"request_module: runaway loop modprobe" (Looping legacy kernel modprobe that applies to older Linux versions, e.g. 2.6.16-xenU)

This condition is indicated by a system log similar to the one shown below. Using an unstable or old Linux kernel can cause an interminable loop condition at startup.

```
Linux version 2.6.16-xenU (builder@xenbat.amazonsa) (gcc version 4.0.1
20050727 (Red Hat 4.0.1-5)) #1 SMP Mon May 28 03:41:49 SAST 2007

BIOS-provided physical RAM map:

 Xen: 0000000000000000 - 0000000026700000 (usable)

0MB HIGHMEM available.
…
request_module: runaway loop modprobe binfmt-464c

request_module: runaway loop modprobe binfmt-464c

request_module: runaway loop modprobe binfmt-464c

request_module: runaway loop modprobe binfmt-464c

request_module: runaway loop modprobe binfmt-464c
```

Suggested Actions

For this instance type	Do this
EBS-backed	Use a newer kernel, either Grub-based or static using one of the two procedures below. • Relaunch the instance and specify the `-kernel` and `-ramdisk` parameters. OR • Stop the instance. • Modify the kernel and ramdisk attributes to use a newer kernel. • Start the instance.
Instance-store backed	Relaunch the instance and specify the `-kernel` and `-ramdisk` parameters.

"FATAL: kernel too old" and "fsck. …: No such file or directory while trying to open /dev" (Kernel and AMI mismatch)

This condition is indicated by a system log similar to the one shown below:

```
Linux version 2.6.16.33-xenU (root@dom0-0-50-45-1-a4-ee.z-2.aes0.internal)
(gcc version 4.1.1 20070105 (Red Hat 4.1.1-52)) #2 SMP Wed Aug 15 17:27:36 SAST
 2007
…
FATAL: kernel too old
Kernel panic - not syncing: Attempted to kill init!
```

Potential Causes

Incompatible kernel and userland.

Suggested Actions

For this instance type	Do this

EBS-backed	1. Modify the instance to use newer kernel.
	   a. Stop the instance.
	   b. Modify the configuration to use a newer kernel.
	   c. Start the instance.
	2. Modify the AMI to use a newer kernel.

Instance-store backed	1. Modify the instance to use newer kernel.
	   a. Stop the instance.
	   b. Modify the configuration to use a newer kernel.
	   c. Start the instance.
	2. Modify the AMI to use a newer kernel.

"FATAL: Could not load /lib/modules" or "BusyBox" (Missing kernel modules)

This condition is indicated by a system log similar to the one shown below.

```
[    0.370415] Freeing unused kernel memory: 1716k freed
Loading, please wait...
WARNING: Couldn't open directory /lib/modules/2.6.34-4-virtual: No such file
or directory
FATAL: Could not open /lib/modules/2.6.34-4-virtual/modules.dep.temp for writing:
 No such file or directory
FATAL: Could not load /lib/modules/2.6.34-4-virtual/modules.dep: No such file
or directory
Couldn't get a file descriptor referring to the console
Begin: Loading essential drivers... ...
FATAL: Could not load /lib/modules/2.6.34-4-virtual/modules.dep: No such file
or directory
FATAL: Could not load /lib/modules/2.6.34-4-virtual/modules.dep: No such file
or directory
Done.
```

```
Begin: Running /scripts/init-premount ...
Done.
Begin: Mounting root file system... ...
Begin: Running /scripts/local-top ...
Done.
Begin: Waiting for root file system... ...
Done.
Gave up waiting for root device.  Common problems:
 - Boot args (cat /proc/cmdline)
   - Check rootdelay= (did the system wait long enough?)
   - Check root= (did the system wait for the right device?)
 - Missing modules (cat /proc/modules; ls /dev)
FATAL: Could not load /lib/modules/2.6.34-4-virtual/modules.dep: No such file
or directory
FATAL: Could not load /lib/modules/2.6.34-4-virtual/modules.dep: No such file
or directory
ALERT! /dev/sda1 does not exist. Dropping to a shell!


BusyBox v1.13.3 (Ubuntu 1:1.13.3-1ubuntu5) built-in shell (ash)
Enter 'help' for a list of built-in commands.

(initramfs)
```

Potential Causes

One or more of the following conditions can cause this problem:

- Missing ramdisk

- Missing correct modules from ramdisk

- EBS root volume not correctly attached as /dev/sda1

Suggested Actions

For this instance type	Do this
EBS-backed	1. Select corrected ramdisk for the EBS volume. 2. Stop the instance. 3. Detach the volume and repair it. 4. Attach the volume to the instance. 5. Start the instance. 6. Modify the AMI to use the corrected ramdisk.
Instance-store backed	1. Terminate the instance and relaunch with correct ramdisk. 2. Rebundle the AMI with the corrected ramdisk definition.

"ERROR Invalid kernel" (EC2 incompatible kernel)

This condition is indicated by a system log similar to the one shown below.

```
...
root (hd0)

 Filesystem type is ext2fs, using whole disk

kernel /vmlinuz root=/dev/sda1 ro

initrd /initrd.img

ERROR Invalid kernel: elf_xen_note_check: ERROR: Will only load images
built for the generic loader or Linux images
xc_dom_parse_image returned -1

Error 9: Unknown boot failure

  Booting 'Fallback'

root (hd0)

 Filesystem type is ext2fs, using whole disk

kernel /vmlinuz.old root=/dev/sda1 ro

Error 15: File not found
```

Potential Causes

One or both of the following conditions can cause this problem:

- Supplied kernel is not supported by Grub.
- Fallback kernel does not exist.

Suggested Actions

For this instance type	Do this
EBS-backed	1. Stop the instance. 2. Replace with working kernel. 3. Install a fallback kernel. 4. Modify the AMI by correcting the kernel.
Instance-store backed	1. Terminate and relaunch the instance with the correct kernel. 2. Rebundle AMI with the corrected kernel definition. 3. (Optional) Seek technical assistance for data recovery using Premium Support.

File System Errors

"request_module: runaway loop modprobe " (Looping legacy kernel modprobe that applies to older Linux versions, e.g. 2.6.16-xenU)

This condition is indicated by a system log similar to the one shown below. Using an unstable or old Linux kernel can cause an interminable loop condition at startup.

```
Linux version 2.6.16-xenU (builder@xenbat.amazonsa) (gcc version 4.0.1
20050727 (Red Hat 4.0.1-5)) #1 SMP Mon May 28 03:41:49 SAST 2007

BIOS-provided physical RAM map:

 Xen: 0000000000000000 - 0000000026700000 (usable)

0MB HIGHMEM available.
…

request_module: runaway loop modprobe binfmt-464c

request_module: runaway loop modprobe binfmt-464c

request_module: runaway loop modprobe binfmt-464c

request_module: runaway loop modprobe binfmt-464c

request_module: runaway loop modprobe binfmt-464c
```

Suggested Actions

For this instance type	Do this
EBS-backed	Use a newer kernel, either Grub-based or static using one of the two procedures below. • Relaunch the instance and specify the –kernel and –ramdisk parameters. OR • Stop the instance. • Modify the kernel and ramdisk attributes to use a newer kernel. • Start the instance.
Instance-store backed	Relaunch the instance and specify the –kernel and –ramdisk parameters.

"fsck. … : No such file or directory while trying to open /dev" (Filesystem not found)

This condition is indicated by a system log similar to the one shown below:

```
  Welcome to Fedora
  Press 'I' to enter interactive startup.
Setting clock : Wed Oct 26 05:52:05 EDT 2011 [  OK  ]

Starting udev: [  OK  ]

Setting hostname localhost:  [  OK  ]

No devices found
Setting up Logical Volume Management: File descriptor 7 left open
  No volume groups found
[  OK  ]

Checking filesystems
Checking all file systems.
[/sbin/fsck.ext3 (1) -- /] fsck.ext3 -a /dev/sda1
/dev/sda1: clean, 82081/1310720 files, 2141116/2621440 blocks
[/sbin/fsck.ext3 (1) -- /mnt/dbbackups] fsck.ext3 -a /dev/sdh
fsck.ext3: No such file or directory while trying to open /dev/sdh

/dev/sdh:
The superblock could not be read or does not describe a correct ext2
filesystem.  If the device is valid and it really contains an ext2
filesystem (and not swap or ufs or something else), then the superblock
is corrupt, and you might try running e2fsck with an alternate superblock:
    e2fsck -b 8193 <device>

[FAILED]


*** An error occurred during the file system check.
*** Dropping you to a shell; the system will reboot
*** when you leave the shell.
Give root password for maintenance
(or type Control-D to continue):
```

Potential Causes

- A bug exists in ramdisk filesystem definitions /etc/fstab

- Misconfigured filesystem definitions in /etc/fstab

- Missing/failed drive

Suggested Actions

For this instance type	Do this
EBS-backed	• Stop the instance, detach the root volume, repair/modify /etc/fstab the volume, attach the volume to the instance, and start the instance. • Fix ramdisk to include modified /etc/fstab (if applicable). • Modify the AMI to use a newer ramdisk. **Tip** The sixth field in the fstab defines availability requirements of the mount – a non-zero value implies that an fsck will be done on that volume and *must* succeed. Using this field can be problematic in Amazon EC2 because a failure typically results in an interactive console prompt which is not currently available in Amazon EC2. Use care with this feature and read the Linux man page for fstab.
Instance-store backed	• Terminate the instance and relaunch another instance. • Detach any errant EBS volumes and the reboot instance. • (Optional) Seek technical assistance for data recovery using Premium Support.

"General error mounting filesystems" (Failed Mount)

This condition is indicated by a system log similar to the one shown below.

```
Loading xenblk.ko module
xen-vbd: registered block device major 8

Loading ehci-hcd.ko module
Loading ohci-hcd.ko module
Loading uhci-hcd.ko module
USB Universal Host Controller Interface driver v3.0

Loading mbcache.ko module
Loading jbd.ko module
Loading ext3.ko module
Creating root device.
Mounting root filesystem.
kjournald starting.  Commit interval 5 seconds

EXT3-fs: mounted filesystem with ordered data mode.
```

```
Setting up other filesystems.
Setting up new root fs
no fstab.sys, mounting internal defaults
Switching to new root and running init.
unmounting old /dev
unmounting old /proc
unmounting old /sys
mountall:/proc: unable to mount: Device or resource busy
mountall:/proc/self/mountinfo: No such file or directory
mountall: root filesystem isn't mounted
init: mountall main process (221) terminated with status 1

General error mounting filesystems.
A maintenance shell will now be started.
CONTROL-D will terminate this shell and re-try.
Press enter for maintenance
(or type Control-D to continue):
```

Potential Causes

Instance type	Potential cause
EBS-backed	• Detached or failed EBS volume. • Corrupted filesystem. • Mismatched ramdisk and AMI combination (e.g., Debian ramdisk with a Suse AMI).
Instance-store backed	• Failed drive. • Corrupted filesystem. • Mismatched ramdisk and combination (e.g., Debian ramdisk with a Suse AMI).

Suggested Actions

For this instance type	Do this
EBS-backed	Use the following procedure: 1. Stop the instance. 2. Detach the root volume. 3. Attach the root volume to a known working instance. 4. Run filesystem check (fsck –a /dev/…). 5. Fix any errors. 6. Detach the volume from the known working instance. 7. Attach the volume to the stopped instance. 8. Start the instance. 9. Recheck the instance status.
Instance-store backed	Try one of the following: • Restart a new instance. • (Optional) Seek technical assistance for data recovery using Premium Support.

"Kernel panic - not syncing: VFS: Unable to mount root fs on unknown-block" (Root filesystem mismatch)

This condition is indicated by a system log similar to the one shown below.

```
Linux version 2.6.16-xenU (builder@xenbat.amazonsa) (gcc version 4.0.1
 20050727 (Red Hat 4.0.1-5)) #1 SMP Mon May 28 03:41:49 SAST 2007
…
Kernel command line:  root=/dev/sda1 ro 4
…
Registering block device major 8
…
Kernel panic – not syncing: VFS: Unable to mount root fs on unknown-block(8,1)
```

Potential Causes

Instance type	Potential cause
EBS-backed	• Device not attached correctly. • Root device not attached at correct device point. • Filesystem not expected format. • Use of legacy kernel (e.g., 2.6.16-XenU).

Instance type	Potential cause
Instance-store backed	Hardware device failure.

Suggested Actions

For this instance type	Do this
EBS-backed	Do one of the following: • Stop and then restart the instance. • Modify root volume to attach at the correct device point, possible /dev/sda1 instead of /dev/sda. • Stop and modify to use modern kernel.
Instance-store backed	Terminate and relaunch the instance. We strongly recommend using a modern kernel.

"Error: Unable to determine major/minor number of root device…" or "[ramfs /]#" (Root filesystem/device mismatch)

This condition is indicated by a system log similar to the one shown below.

```
...
XENBUS: Device with no driver: device/vif/0
XENBUS: Device with no driver: device/vbd/2048
drivers/rtc/hctosys.c: unable to open rtc device (rtc0)
Initializing network drop monitor service
Freeing unused kernel memory: 508k freed
:: Starting udevd...
done.
:: Running Hook [udev]
:: Triggering uevents...<30>udevd[65]: starting version 173
done.
Waiting 10 seconds for device /dev/xvda1 ...
Root device '/dev/xvda1' doesn't exist. Attempting to create it.
ERROR: Unable to determine major/minor number of root device '/dev/xvda1'.
You are being dropped to a recovery shell
    Type 'exit' to try and continue booting
sh: can't access tty; job control turned off
[ramfs /]#
```

Potential Causes

• Missing or incorrectly configured virtual block device driver

• Device enumeration clash (sda versus xvda or sda instead of sda1)

• Incorrect choice of DomU kernel

Suggested Actions

For this instance type	Do this
EBS-backed	1. Stop the instance. 2. Detach the volume. 3. Fix the device mapping problem. 4. Start the instance. 5. Modify the AMI to address device mapping issues.
Instance-store backed	1. Rebundle with the appropriate fix (map block device correctly). 2. Terminate and relaunch the instance.

"XENBUS: Device with no driver: device/vif/0""

This condition is indicated by a system log similar to the one shown below.

```
XENBUS: Device with no driver: device/vbd/2048
drivers/rtc/hctosys.c: unable to open rtc device (rtc0)
Initalizing network drop monitor service
Freeing unused kernel memory: 508k freed
:: Starting udevd...
done.
:: Running Hook [udev]
:: Triggering uevents...<30>udevd[65]: starting version 173
done.
Waiting 10 seconds for device /dev/xvda1 ...
Root device '/dev/xvda1' doesn't exist. Attempting to create it.
ERROR: Unable to determine major/minor number of root device '/dev/xvda1'.
You are being dropped to a recovery shell
    Type 'exit' to try and continue booting
sh: can't access tty; job control turned off
[ramfs /]#
```

Potential Causes

- Missing or incorrectly configured virtual block device driver
- Device enumeration clash (sda versus xvda)
- Incorrect choice of DomU kernel

Suggested Actions

For this instance type	Do this
EBS-backed	1. Stop the instance. 2. Detach the volume. 3. Fix the device mapping problem. 4. Start the instance. 5. Modify the AMI to address device mapping issues.
Instance-store backed	1. Rebundle with the appropriate fix (map block device correctly). 2. Terminate and relaunch the instance.

Filesystem check required: "… days without being checked, check forced."

This condition is indicated by a system log similar to the one shown below.

```
...
Checking filesystems
Checking all file systems.
[/sbin/fsck.ext3 (1) -- /] fsck.ext3 -a /dev/sda1
/dev/sda1 has gone 361 days without being checked, check forced
```

Potential Causes

Filesystem check time passed; a filesystem check is being forced.

Suggested Actions

- Wait until the filesystem check completes. Note that a filesystem check can take a long time depending on the size of the root filesystem.
- Modify your filesystems to remove the filesystem check (fsck) enforcement using tune2fs or tools appropriate for your filesystem.

"fsck died with exit status…" (Missing device).

This condition is indicated by a system log similar to the one shown below.

```
Cleaning up ifupdown....
Loading kernel modules...done.
…
Activating lvm and md swap...done.
Checking file systems...fsck from util-linux-ng 2.16.2
/sbin/fsck.xfs: /dev/sdh does not exist
fsck died with exit status 8
[31mfailed (code 8).[39;49m
```

Potential Causes

- Ramdisk looking for missing drive
- Filesystem consistency check forced
- Drive failed or detached

Suggested Actions

For this instance type	Do this
EBS-backed	Try one or more of the following to resolve the issue: • Stop the instance, attach volume to existing running instance. • Manually run consistency checks. • Fix ramdisk to include relevant utilities. • Modify filesystem tuning parameters to remove consistency requirements (not recommended).
Instance-store backed	Try one or more of the following to resolve the issue: • Rebundle ramdisk with correct tooling. • Modify filesystem tuning parameters to remove consistency requirements (not recommended). • Terminate and restart. • (Optional) Seek technical assistance for data recovery using Premium Support.

Operating System Errors

GRUB prompt: "grubdom>")

This condition is indicated by a system log similar to the one shown below.

```
    GNU GRUB   version 0.97  (629760K lower / 0K upper memory)

      [ Minimal BASH-like line editing is supported.   For

        the   first   word,  TAB  lists  possible  command

        completions.   Anywhere else TAB lists the possible

        completions of a device/filename. ]

grubdom>
```

Potential Causes

Instance type	Potential causes
EBS-backed	• Missing grub.conf file. • Incorrect Grub image used expecting grub.conf at different location. • Unsupported filesystem used to store grub.conf.
Instance-store backed	• Missing grub.conf file. • Incorrect Grub image used expecting grub.conf at different location.

Suggested Actions

For this instance type	Do this
EBS-backed	**Option 1: Modify the AMI and relaunch the instance** 1. Modify the source AMI to create a grub.conf at the standard location (/boot/grub/menu.lst). 2. Pick the appropriate Grub image, (hd0-1st drive or hd00 – 1st drive, 1st partition). 3. Relaunch the instance. **Option 2: Fix the existing instance** 1. Stop the instance. 2. Detach the root filesystem. 3. Attach the root filesystem to a known working instance. 4. Mount filesystem. 5. Create grub.conf. 6. Detach filesystem. 7. Attach to the original instance. 8. Modify kernel attribute to use appropriate Grub (1st disk or 1st partition on 1st disk). 9. Start the instance.

For this instance type	Do this
Instance-store backed	**Option 1: Modify the AMI and relaunch the instance** 1. Modify the source AMI to create a grub.conf at the standard location (/boot/grub/menu.lst). 2. Pick the appropriate Grub image, (hd0-1st drive or hd00 – 1st drive, 1st partition). 3. Relaunch the instance. **Option 2: Fix the existing instance** Relaunch the instance using the correct kernel if you believe the kernel was incorrect. **Note** To recover data from the existing instance, contact AWS Premium Support.

"Bringing up interface eth0: Device eth0 has different MAC address than expected, ignoring. [FAILED]" (Hardcoded MAC address)

This condition is indicated by a system log similar to the one shown below:

```
...
Bringing up loopback interface:  [  OK  ]

Bringing up interface eth0:  Device eth0 has different MAC address than expected,
 ignoring.
[FAILED]

Starting auditd: [  OK  ]
```

Potential Causes

There is a hardcoded interface MAC in the AMI configuration.

Suggested Actions

For this instance type	Do this
EBS-backed	**Option 1** • Modify the AMI to remove the hard coding and relaunch the instance. **Option 2** • Modify the instance to remove the hard coded MAC address. **Option 3** 1. Stop the instance. 2. Detach the root filesystem. 3. Modify the instance to remove the hard coded MAC address. 4. Attach the volume to original instance. 5. Start the instance.
Instance-store backed	• Modify the instance to remove the hard coded MAC address. • Relaunch the instance.

"Unable to load SELinux Policy. Machine is in enforcing mode. Halting now. … Kernel panic - not syncing: Attempted to kill init!" (SELinux misconfiguration)

This condition is indicated by a system log similar to the one shown below.

```
audit(1313445102.626:2): enforcing=1 old_enforcing=0 auid=4294967295
Unable to load SELinux Policy. Machine is in enforcing mode. Halting now.
Kernel panic - not syncing: Attempted to kill init!
```

Potential Causes

SELinux has been enabled in error:

• Supplied kernel is not supported by Grub.
• Fallback kernel does not exist.

Suggested Actions

For this instance type	Do this
EBS-backed	1. Stop the instance. 2. Detach the volume. 3. Disable SELinux. 4. Start the instance.
Instance-store backed	1. Terminate and relaunch the instance. 2. (Optional) Seek technical assistance for data recovery using Premium Support.

" XENBUS: Timeout connecting to devices!" ((Xenbus timeout))

This condition is indicated by a system log similar to the one shown below.

```
Linux version 2.6.16-xenU (builder@xenbat.amazonsa) (gcc version 4.0.1
20050727 (Red Hat 4.0.1-5)) #1 SMP Mon May 28 03:41:49 SAST 2007
…
XENBUS: Timeout connecting to devices!
…
Kernel panic - not syncing: No init found.  Try passing init= option to kernel.
```

Potential Causes

- The block device not is connected to the instance.
- This instance is using a very old DomU kernel.

Suggested Actions

For this instance type	Do this
EBS-backed	- Modify AMI and instance to use a modern kernel and relaunch the instance. - Reboot the instance.
Instance-store backed	- Reboot the instance. - Modify AMI and instance to use a modern kernel and relaunch the instance.

What to Do If an Instance Immediately Terminates

We recommend that when you launch an instance, you immediately check its status to confirm that it goes to the *running* status and not *terminated*.

Why would an Amazon EBS-backed instance immediately terminate? Following are a few reasons why:

- You've reached your volume limit. Your account is limited to 5000 volumes, or 20 TiB in total volume storage, whichever you reach first. You can request to increase your volume limit by completing the Amazon EBS Volume Limit Request Form.
- The AMI is missing a required part.
- The snapshot is corrupt.

This section describes how to get information about why the instance terminated.

AWS Management Console

To get information about why the instance terminated

1. Go to the **Instances** page in the console and view the instance details.

2. Look at **State Transition Reason**.

   The field shows information about why the instance terminated. For a normal instance, the field is usually blank or shows *User initiated* if you've requested to stop or start the instance.

Command Line Tools

To get information about why the instance terminated

1. Use the **ec2-describe-instances** command in verbose mode:

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

When you use verbose mode, the response includes the underlying SOAP request and the SOAP XML response.

2. In the XML response that's displayed, locate the `stateReason` element. It looks similar to the following example.

```
<stateReason>
    <code>Client.UserInitiatedShutdown</code>
    <message>Client.UserInitiatedShutdown: User initiated shutdown</message>
</stateReason>
```

The preceding example shows what's displayed for a normal instance that you've stopped. For the instance that terminated immediately, the `code` and `message` elements will describe the reason for the termination (e.g., `VolumeLimitExceeded`).

API

To get information about why the instance terminated

1. Construct the following Query request. For information about the auth parameters, go to Common Query Parameters in the *Amazon Elastic Compute Cloud API Reference*.

```
https://ec2.amazonaws.com/
?Action=DescribeInstances
&AUTHPARAMS
```

2. In the XML response that's displayed, locate the `stateReason` element. It looks similar to the following example.

```
<stateReason>
    <code>Client.UserInitiatedShutdown</code>
    <message>Client.UserInitiatedShutdown: User initiated shutdown</message>
</stateReason>
```

The preceding example shows what's displayed for a normal instance that you've stopped. For the instance that terminated immediately, the `code` and `message` elements will describe the reason for the termination (e.g., `VolumeLimitExceeded`).

Getting Console Output and Rebooting 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 API and the command line interface tools (CLI).

Console Output

For Linux/UNIX instances, 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 instance's owner can retrieve it.

For Windows instances, the Amazon EC2 instance console output displays the last three system event log errors.

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 using `GetConsoleOutput`. For more information, go to the Amazon Elastic Compute Cloud API Reference or Amazon Elastic Compute Cloud Command Line Reference.

Note

Only the instance owner can access the console output.

Instance Reboot

Just as you can reset a machine by pressing the reset button, you can reset Amazon EC2 instances using `RebootInstances`. For more information, go to the Amazon Elastic Compute Cloud API Reference or Amazon Elastic Compute Cloud Command Line Reference.

Caution

For Windows instances, this operation performs a hard reboot that might result in data corruption.

Network and Security

Topics

This section describes key network and security features related to Amazon EC2.

Amazon EC2 Credentials

Topics

Types of Credentials You Need

Amazon EC2 uses a variety of credentials for different purposes. This section describes major tasks you might perform with EC2 and the credentials required to perform them.

Credentials to Access Your Account

To sign up for services, view your bills, perform account-based tasks, and get many of your security credentials, you will need your standard Amazon login and password. For more information, see How to Log In with Your Amazon Login and Password (p. 351).

Note

We also provide AWS Multi-Factor Authentication, which requires a physical device and passcode to login to your AWS account. For more information, go to http://aws.amazon.com/mfa.

Credentials to Launch and Administer Amazon EC2 Instances

The credentials you need to launch and administer instances depend on the interface you're using.

AWS Management Console

To launch and administer Amazon EC2 instances through the AWS Management Console, you only need the Amazon login and password. For more information, see How to Log In with Your Amazon Login and Password (p. 351).

Query and Third-Party UI Tools

Launching and administering Amazon EC2 instances through the Query API and many UI-based tools (e.g., ElasticFox) requires the Access Key ID and Secret Access Key. For more information, see How to Get Your Access Key ID and Secret Access Key (p. 352).

SOAP and EC2 Command Line Tools

Launching and administering Amazon EC2 instances through the SOAP API and command line interface (i.e., API tools) requires an X.509 certificate and private key, which can be generated by AWS. For more information, see How to Create an X.509 Certificate and Private Key (p. 352).

Credentials to Connect to Amazon EC2 Instances

To connect to your instances, you need the following:

- **Amazon EC2 Key Pair—**Enables you to connect to Linux/UNIX instances through SSH. For more information, see SSH Key Pair (p. 353).
- **Windows administrator password (Windows only)—**Provides the "first-use" password that enables you to connect to a Windows instance through Remote Desktop. For more information, see Windows Administrator Password (p. 353).

Credentials Needed for Sharing with Others

To enable other AWS accounts to use your Amazon EC2 AMIs and Amazon EBS snapshots, you need their AWS Account IDs. For information on how to get the AWS Account ID associated with your account, see Viewing Your Account ID (p. 353).

Credentials Needed for Bundling Amazon EC2 instance store-backed Instances

To bundle Amazon EC2 instance store-backed Linux/UNIX instances, you need your AWS Account ID, and your X.509 certificate and private key. For information about viewing your Account ID, see Viewing Your Account ID (p. 353). For information about getting an X.509 certificate and private key, see How to Create an X.509 Certificate and Private Key (p. 352).

To bundle Amazon EC2 instance store-backed Windows instances, you need your Amazon login and password to access the AWS Management Console. For information about the Amazon login and password, see How to Log In with Your Amazon Login and Password (p. 351).

How to Log In with Your Amazon Login and Password

The Amazon login and password enable you to sign up for services, view your bills, perform account-based tasks, and get many of your security credentials. You also use the login and password to perform Amazon EC2 tasks through the AWS Management Console.

This section describes how to log in with your login and password.

To log in with your login and password (if you have an existing account)

1. Go to the AWS Web Site.

2. Select an option from the Your Account menu. The **Amazon Web Services Sign In** page appears.

3. Enter your email address, select I am a returning user and my password is, enter your password, and click the **Sign In** button.

To get a new Amazon login and password (create a new AWS account)

1. Go to the AWS Web Site.

2. Click **Create an AWS Account**.

   The **Amazon Web Services Sign In** page appears.

3. Select I am a new user and click the **Sign In** button.

4. Follow the on-screen prompts to create a new account.

Note

It is important to keep your Amazon login and password secret as they can be used to view and create new credentials. As an increased security measure, we offer Multi-Factor Authentication, which uses the combination of a physical device and passcode to login to your AWS account. For more information, go to http://aws.amazon.com/mfa.

How to Get Your Access Key ID and Secret Access Key

The Access Key ID and Secret Access Key are the most commonly used set of AWS credentials. They are used to make Query and REST-based requests and are commonly used by UI-based tools, such as ElasticFox. You can use up to two sets of Access Keys at a time. You can generate new keys at any time or disable existing keys.

To get your Access Key ID and Secret Access Key

1. Go to the AWS Web Site.

2. Point to **Your Account** and select **Security Credentials**.

   If you are not already logged in, you are prompted to do so.

3. Scroll down to the **Access Credentials** section and verify the **Access Keys** tab is selected.

4. Locate an active Access Key in the **Your Access Keys** list.

5. To display the Secret Access Key, click **Show** in the **Secret Access Key** column.

6. Write down the keys or save them.

7. If there are no Access Keys in the list, click **Create a New Access Key** and follow the on-screen prompts.

How to Create an X.509 Certificate and Private Key

The X.509 Certificate and Private Key are used by the command line tools and the SOAP API.

If you have and are using the AWS Identity and Access Management (IAM) account, you will have to first create the certificate and then upload it to the IAM system before you can use it. Currently, IAM does not have API action to create signing certificate, so you must use a third-party tool such as OpenSSL to create the certificate first. For detailed instructions on creating and uploading a signing certificate, go to Creating and Uploading Server Certificates.

This section describes how to create a new certificate using your AWS account. You can download the private key file once. If you lose it, you will need to create a new certificate. Up to two certificates (active or inactive) are allowed at any time.

To create a certificate using your AWS account

1. Go to the AWS Web Site.

2. Point to **Your Account** and select **Security Credentials**.

   If you are not already logged in, you are prompted to do so.

3. Click the **X.509 Certificates** tab

4. Click **Create a New Certificate** and follow the on-screen prompts.

Note

If you already have two certificates, you will not see the **Create a New Certificate** option. If you want to create a new certificate, delete one of your existing certificates, after ensuring you are not using it. When the deletion is complete, you will see the **Create a New Certificate** option, which indicates that you can create an additional certificate.

The new Certificate is created and appears in the X.509 Certificate list. You are prompted to download the certificate and private key files.

5. Create a .ec2 directory in your home directory, and save these files to it with the filenames offered by your browser.

   You should end up with a PEM-encoded X.509 certificate and a private key file.

SSH Key Pair

You must create an RSA public/private key pair, which you use to ensure that only you have access to instances that you launch.

You have two options for getting this key pair:

- Generate it yourself with a third-party tool such as OpenSSH, and then import the public key to AWS using either the `ec2-import-keypair` command or the `ImportKeyPair` action

- Have AWS generate the key pair for you using the AWS Management Console, the `ec2-add-keypair` command, or the `CreateKeyPair` action

With either option, AWS doesn't store a copy of the private key. Amazon EC2 only stores the public key, and associates it with a friendly key pair name you specify. 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.

For information on how to create key pairs, see Getting an SSH Key Pair (p. 255).

Windows Administrator Password

The Windows administrator password is used to access a Windows instance through Remote Desktop (RDP) for the first time only. If you change the password or rebundle the AMI, the instance will use the last set password. For information on how to get the Windows administrator password to connect to a Windows instance through RDP using the AWS Management Console, see Connecting to Windows Instances (p. 275).

Viewing Your Account ID

The Account ID identifies your account to AWS and enables other accounts to access resources that you want to share, such as Amazon EC2 AMIs and Amazon EBS snapshots.

To view your Account ID

1. Go to the AWS Web Site.

2. Point to **Your Account** and select **Security Credentials**.

   If you are not already logged in, you are prompted to do so.

3. Scroll down to the **Account Identifiers** section.

4. Locate your AWS Account ID.

For information on how to share AMIs, see Sharing AMIs Safely (p. 62). For information on how to share snapshots, see Modifying Snapshot Permissions (p. 468).

Note

The Account ID number is not a secret. When granting access to resources, make sure to specify the Account ID without hyphens.

The canonical user ID is used by Amazon S3.

Related Topics

*   ???
*   Launching Amazon EC2 Instances (p. 217)
*   Connecting to Amazon EC2 Instances (p. 255)

Amazon Virtual Private Cloud

Amazon Virtual Private Cloud enables you to create a virtual network in the AWS cloud. With a Virtual Private Cloud (VPC), you can define a virtual network that closely resembles a traditional data center. You have complete control over your virtual networking environment, including selection of your own IP address range, creation of subnets, and configuration of routing and access control lists. This section gives a brief introduction to Amazon VPC.

Amazon Virtual Private Cloud Concepts

If you're familiar with Amazon EC2, you know that each instance you launch is randomly assigned a public IP address in the Amazon EC2 address space. Amazon VPC enables you to create an isolated portion of the AWS cloud (a *VPC*) and launch Amazon EC2 instances that have private (RFC 1918) addresses in the range of your choice (e.g., 10.0.0.0/16). You can define *subnets* within your VPC, which enable you to group similar kinds of instances based on IP address range.

By using Amazon VPC with Amazon EC2 (instead of Amazon EC2 alone), you gain the ability to:

- Logically group your Amazon EC2 instances, and assign them private IP addresses
- Control the egress traffic from your Amazon EC2 instances (in addition to controlling the ingress traffic to them)
- Add an additional layer of security to your Amazon EC2 instances in the form of network Access Control Lists (ACLs)
- Connect your VPC to your corporate data center with a VPN connection, so you can use the AWS cloud as an extension of your corporate data center network

Levels of Privacy

When you create a VPC, you can configure it based on the level of privacy you want. In the most private scenario, you can attach only a *virtual private gateway*, and create an IPsec tunnel between your VPC and home network. In this scenario, your EC2 instances have no direct exposure to the Internet.

In the most public scenario, you can attach only an *Internet gateway* to the VPC and enable traffic to flow between the Internet and all the instances in your VPC.

You can configure your VPC to be somewhere in between, with both a virtual private gateway and an Internet gateway. Here, some instances could receive Internet traffic (e.g., web servers), whereas others could remain unexposed (e.g., database servers). This is a common scenario for running a multi-tier web application in the AWS cloud.

These different scenarios are discussed in more detail in the Amazon VPC documentation.

Routing and Security

You can configure routing in your VPC to control where traffic flows (e.g., to the Internet gateway, virtual private gateway, etc). With an Internet gateway, your VPC has direct access to other AWS products such as Amazon Simple Storage Service (Amazon S3). If you choose to have only a virtual private gateway with a connection to your home network, you can route your Internet-bound traffic over the VPN and control its egress with your security policies and corporate firewall. In the latter case, you incur additional bandwidth charges when accessing AWS products over the Internet.

You can use *security groups* and *network ACLs* to help secure the instances in your VPC. Security groups might be familiar if you're an Amazon EC2 user, and network ACLs might be familiar if you're a network administrator. Security groups act like a firewall at the instance level, whereas network ACLs are an additional layer of security that act at the subnet level.

By default, the instances you launch in your VPC have only private IP addresses. If you want an instance to have a public IP address, you can assign it an *Elastic IP address*, which is a static, public address you can assign to any instance in your VPC. For an instance in your VPC to be addressable from the Internet, it must have an Elastic IP address.

You can use Network Address Translation (NAT) to enable instances that don't have Elastic IP addresses to reach the Internet. You can set up the VPC's routing so that traffic from private instances goes through a special NAT instance that has an Elastic IP address. We provide a NAT Amazon Machine Image (AMI) that you can use for this purpose.

Dedicated Instances

Amazon EC2 instances launched into a VPC have a tenancy attribute. Setting the instance's tenancy attribute to `dedicated` specifies that your instance will run on single-tenant hardware. Amazon VPCs have a related attribute called *instance tenancy*. Setting this instance tenancy attribute to `dedicated` specifies that only Dedicated Instances can be launched into the VPC.

For more information, go to Using EC2 Dedicated Instances Within Your VPC in the *Amazon Virtual Private Cloud User Guide*.

Amazon VPC Documentation

Amazon VPC has its own set of documentation to describe how to create and use your VPC. The following table gives links to the Amazon VPC guides.

Description	Documentation
How to get started using Amazon VPC	Amazon Virtual Private Cloud Getting Started Guide
How to use Amazon VPC through the AWS Management Console	Amazon Virtual Private Cloud User Guide
Complete descriptions of all the Amazon VPC commands	Amazon Elastic Compute Cloud Command Line Reference (the Amazon VPC commands are part of the Amazon EC2 reference)
Complete descriptions of the Amazon VPC API actions, data types, and errors	Amazon Elastic Compute Cloud API Reference (the Amazon VPC API actions are part of the Amazon EC2 reference)
Information for the network administrator who needs to configure the gateway at your end of an optional IPsec VPN connection	Amazon Virtual Private Cloud Network Administrator Guide

AWS Identity and Access Management

Topics

This section provides an introduction to using AWS Identity and Access Management with Amazon Elastic Compute Cloud.

Identity and Access Management

AWS Identity and Access Management (IAM) is a web service that enables Amazon Web Services (AWS) customers to manage users and user permissions in AWS. The service is targeted at organizations with multiple users or systems that use AWS products such as Amazon EC2 and the AWS Management Console. With IAM, you can centrally manage users, security credentials, such as access keys, and permissions that control which AWS resources users can access. For more information on AWS Identity and Access Management, go to Using IAM.

Amazon EC2 Permissions

Amazon EC2 has its own permissions system that covers Amazon Machine Images (AMIs) and Amazon EBS snapshots. There's no ACL system or policy system to give permissions to launch AMIs or create volumes from snapshots. Instead, the Amazon EC2 API lets you modify the attributes on an AMI or snapshot to give another AWS Account those permissions. The following diagram illustrates the concept. Each AMI has a *LaunchPermission* attribute that you can set to one or more AWS Account IDs in order to share the AMI with those AWS Accounts. Each Amazon EBS snapshot has a similar *VolumePermission* attribute.
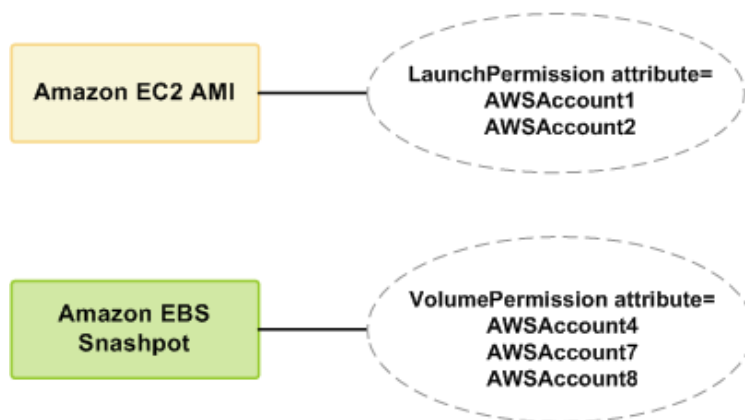
This sharing works at the AWS Account level only; you can't restrict access only to specific users within the AWS Account you're sharing, or only to specific users in your own AWS Account. All users in the AWS Account you're sharing with can use the AMI or snapshot you've shared.

> **Note**
>
> Don't be confused by the fact that the attribute for specifying the AWS Account to share with is called `UserId`. The value you specify for `UserId` is an AWS Account ID.

For information on modifying attributes on an AMI, see Sharing AMIs (p. 66). For information on modifying attributes on a snapshot, see Modifying Snapshot Permissions (p. 468)

Amazon EC2 Permissions and AWS Identity and Access Management (IAM)

Using IAM with Amazon EC2 doesn't change how you use the Amazon EC2 API to share AMIs and snapshots with other AWS Accounts. However, you can use IAM policies to specify which Amazon EC2 actions a user in your AWS Account can use with EC2 resources in general. You can't specify a particular

Amazon EC2 resource in the policy (e.g,. a specific AMI). Instead, you must specify * as the resource to indicate all resources in the AWS Account.

Example 1: Creating a policy

You could create a policy that gives the Developers group permission to use only `RunInstances`, `StopInstances`, `StartInstances`, `TerminateInstances`, and `DescribeInstances`. They could then use those with any AMI that belongs to your AWS Account, any public AMIs, or any AMIs that have been shared with your AWS Account. The following diagram illustrates the concept.

IAM Policy

Allow

Actions:
　　RunInstances, StopInstances,
　　StartInstances, TerminateInstances,
　　DescribeInstances

Resource:　*

Group:
Developers

Important

Amazon EC2 uses SSH keys, Windows passwords, and security groups to control who has access to specific Amazon EC2 instances. You can't use the IAM system to allow or deny access to a specific instance.

Example 2: Setting LaunchPermission

This example builds on the previous one. In addition to the IAM policy attached to the Developers group, you set the `LaunchPermission` attribute on the AMI in your AWS Account with ID ami-12345678 so that two other AWS Accounts can launch the AMI. Anyone possessing the keys for either of those AWS Accounts can launch an instance of the AMI. Also, any user in the Developers group can launch an instance of that AMI, because the Developers group has permission to use `RunInstances` with any AMI in the AWS Account.
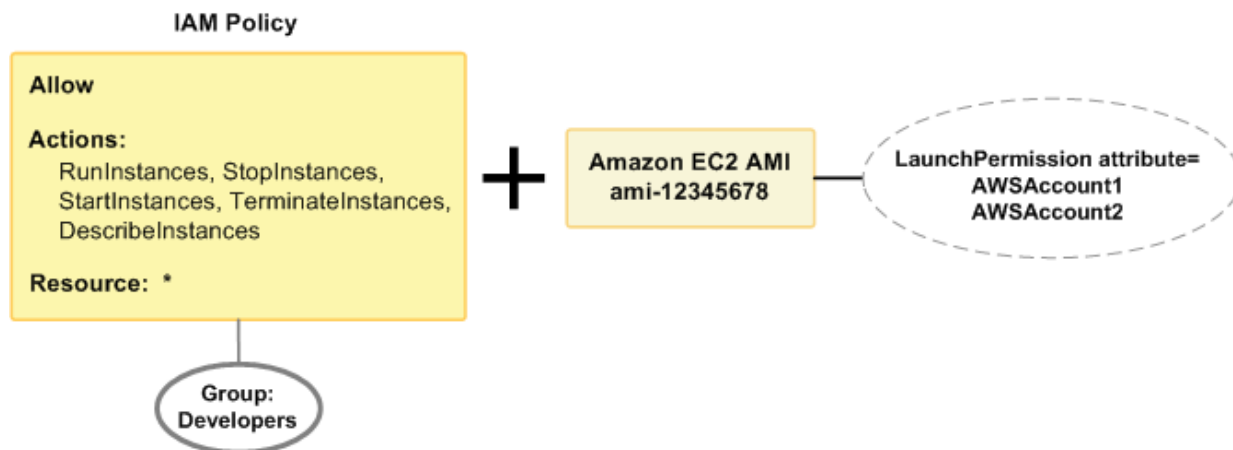
For examples of IAM policies that cover Amazon EC2 actions, see Example Policies for Amazon EC2 (p. 360). For more information about granting permissions to AMIs and snapshots, refer to the topics about the `ModifyImageAttribute` and `ModifySnapshotAttribute` APIs in the Amazon Elastic Compute Cloud API Reference.

No Amazon Resource Names(ARNs) for Amazon EC2

Amazon EC2 has no Amazon Resource Names (ARNs) because you can t specify a particular Amazon EC2 resource in an IAM policy. When writing a policy to control access to Amazon EC2 actions, you use * as the resource. For more information about ARNs, see Identifiers For IAM Entities.

Amazon EC2 Actions

In an IAM policy, you can specify any and all actions that Amazon EC2 offers. Each action name must be prefixed with the lowercase string `ec2:`. For example: `ec2:RunInstances`, `ec2:CreateImage`, `ec2:*` (for all Amazon EC2 actions). For a list of the actions, refer to the Query API or SOAP API action names in the Amazon Elastic Compute Cloud API Reference.

Amazon EC2 Keys

Amazon EC2 implements the following policy keys:

AWS-Wide Policy Keys

- `aws:CurrentTime` (for date/time conditions)
- `aws:EpochTime` (the date in epoch or UNIX time, for use with date/time conditions)
- `aws:SecureTransport` (Boolean representing whether the request was sent using SSL)
- `aws:SourceIp` (the requester's IP address, for use with IP address conditions)
- `aws:UserAgent` (information about the requester's client application, for use with string conditions)

If you use `aws:SourceIp`, and the request comes from an Amazon EC2 instance, we evaluate the instance's public IP address to determine if access is allowed.

For services that use only SSL, such as Amazon RDS and Amazon Route 53, the `aws:SecureTransport` key has no meaning.

The key names are case insensitive. For example, `aws:CurrentTime` is equivalent to `AWS:currenttime`.

Example Policies for Amazon EC2

This section shows several simple policies for controlling user access to Amazon EC2.

> **Note**
>
> In the future, Amazon EC2 might add new actions that should logically be included in one of the following policies, based on the policy's stated goals.

Example 1: Allow a group to only be able to describe, run, stop, start, and terminate instances

In this example, we create a policy that gives access to the relevant actions and attach it to the group. The resource is stated as "*", because you can't specify a particular Amazon EC2 resource in an IAM policy.

```
{
   "Statement":[{
      "Effect":"Allow",
      "Action":["ec2:DescribeInstances","ec2:RunInstances",
                "ec2:StopInstances","ec2:StartInstances",
                "ec2:TerminateInstances"],
      "Resource":"*"
      }
   ]
}
```

Example 2: Allow managers to only be able to list the current Amazon EC2 resources in the AWS Account

In this example, we create a policy that lets managers use the Amazon EC2 actions with `Describe` in the name.

```
{
   "Statement":[{
      "Effect":"Allow",
      "Action":"ec2:Describe*",
      "Resource":"*"
      }
   ]
}
```

Example 3: Share an AMI with a partner

You can't use IAM policies to share a particular AMI with a partner; however, you can do that directly through the Amazon EC2 API. The partner needs his or her own AWS Account. You simply use the Amazon EC2 `ModifyImageAttribute` action or `ec2-modify-image-attribute` command to specify the AWS Account ID you want to share the AMI with. For more information about `ModifyImageAttribute`, go to the Amazon Elastic Compute Cloud API Reference. For more information about `ec2-modify-image-attribute`, go to the Amazon Elastic Compute Cloud Command Line Reference.

Using Temporary Security Credentials

In addition to creating IAM users with their own security credentials, IAM also enables you to grant temporary security credentials to any user allowing this user to access your AWS services and resources. You can manage users who have AWS accounts; these users are IAM users. You can also manage users for your system who do not have AWS accounts; these users are called federated users. Additionally, "users" can also be applications that you create to access your AWS resources.

You can use these temporary security credentials in making requests to Amazon EC2. The API libraries compute the necessary signature value using those credentials to authenticate your request. If you send requests using expired credentials Amazon EC2 denies the request.

For more information about IAM support for temporary security credentials, go to Granting Temporary Access to Your AWS Resources in *Using IAM*.

Example Using Temporary Security Credentials to Authenticate an Amazon EC2 Request

The following example demonstrates how to obtain temporary security credentials to authenticate an Amazon EC2 request.

```
https://ec2.amazonaws.com/?Action=DescribeInstances
&InstanceId.0=I-45fa2e72
&Signature=Dqlp3Sd6ljTUA9Uf6SGtEExwUQEXAMPLE
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2010-03-31T12%3A00%3A00.000Z
&SecurityToken=Security Token Value
&AWSAccessKeyId=Access Key ID provided by AWS Security Token Service
```

Using IAM Roles with Amazon EC2 Instances

Topics

- Launch an EC2 Instance with an IAM Role (p. 362)

Applications running on Amazon Elastic Compute Cloud (Amazon EC2) instances that make requests to Amazon Web Services (AWS) must sign all AWS API requests with AWS access keys that you provide. If you previously distributed your AWS access keys to EC2 instances manually, you can use AWS Identity and Access Management (IAM) roles for EC2 instances instead.

Using IAM roles for EC2 instances, you can securely distribute AWS access keys to EC2 instances and define permissions that applications on those instances use when accessing other services in AWS. For example, you can use IAM roles to grant permissions to an application on an EC2 instance that needs to access Amazon DynamoDB. For more information about creating and using IAM roles, see Working with Roles in Using IAM.

Here are some things you should know about using IAM roles for EC2 instances:

- AWS access keys for signing requests to other services in AWS are automatically made available on running EC2 instances.
- AWS access keys on an EC2 instance are rotated automatically multiple times a day. New access keys will be made available at least five minutes prior to the expiration of the old access keys.
- You can assign granular service permissions for applications running on an EC2 instance that make requests to other services in AWS.

- You can include an IAM role with you launch EC2 On-Demand, Spot, or Reserved Instances.
- IAM roles can be used with all Windows and Linux AMIs.

Warning

If you are using services that use instance metadata service (IMDS) with IAM roles, you should ensure that you do not expose your credentials when the services make HTTP calls on your behalf. You should either include logic to ensure that these services cannot leak information from IMDS, or you should have the appropriate firewall rules in place so that the services cannot access IMDS. Types of services that could expose your credentials include:

- HTTP proxies
- HTML/CSS validator services
- XML processors that support XML inclusion

Launch an EC2 Instance with an IAM Role

You can launch an EC2 instance with an IAM role using the AWS Management Console or by using both the IAM and the EC2 command line interfaces (CLIs). You must create roles in IAM before you can launch an EC2 instance with an IAM role. For more information about creating and using IAM roles, see Working with Roles in Using IAM. You can use the same role for as many instances as you want. After you launch the instance with an IAM role, AWS security credentials are made available through the instance metadata service (IMDS). For more information about instance metadata, see Instance Metadata (p. 302).

Note

After you create an IAM role or instance profile, it may take several seconds for the appropriate permissions to propagate. If your first attempt to launch an EC2 instance with a role fails, you may need to wait a few seconds before trying again. To troubleshoot issues when using IAM roles with EC2 instances, see Troubleshooting Working with Roles in *Using IAM*.

To launch an EC2 instance with an IAM role using the AWS Management Console

1. Create an IAM role. For more information, see Creating a Role in *Using IAM*.
2. Launch an instance with the IAM role. For more information, see Launching an Instance from an AMI (p. 217).
3. Sign AWS requests with the AWS security credentials made available on the EC2 instance. The security credentials that are made available from instance metadata are temporary security credentials. For information on using temporary security credentials, see Using Temporary Security Credentials (p. 361). This is handled for you if you develop your application with the AWS SDK.

   If you are not using the AWS SDK, then you need to programmatically retrieve the AWS security credentials to sign your service API requests for AWS. The following example shows AWS security credentials returned for an IAM role named **s3access**.

```
GET http://169.254.169.254/latest/meta-data/iam/security-credentials/s3access;
 echo {
    "Code" : "Success",
    "LastUpdated" : "2012-04-26T16:39:16Z",
    "Type" : "AWS-HMAC",
    "AccessKeyId" : "AKIAIOSFODNN7EXAMPLE...",
    "SecretAccessKey" : "UtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY...",
    "Token" : "...",
    "Expiration" : "2012-04-26T22:39:16Z"
```

```
    }
```

To launch an EC2 instance with an IAM role using the IAM and EC2 CLIs

You can use the IAM and EC2 CLIs to create a role, add a policy to the role, create an associated instance priofile, and then launch an instance with that instance profile.

1. Create an IAM role, add a policy to it, and create an associated IAM instance profile. The following example shows the creation of an IAM role named **s3access** with a policy that allow the role to access an S3 bucket, and the creation of an instance profile named **s3access**.

    For more information about using the IAM CLI, see iam-rolecreate, iam-roleaddpolicy, and iam-instanceprofilecreate in the *AWS Identity and Access Management (IAM) Command Line Reference*.

    ```
    $ iam-rolecreate -r s3access -s ec2.amazonaws.com

    $ iam-roleaddpolicy -r s3access -e Allow -a s3:\* -c \* -p s3star -o
    {"Version":"2008-10-17","Statement":[{"Effect":"Allow","Action":["s3:*"],"Re
    source":["*"]}]}

    $ iam-instanceprofilecreate -s s3access -r s3access
    arn:aws:iam::111111111111:instance-profile/s3access
    ```

2. Launch an EC2 instance using the instance profile. The following example shows a t1.micro instance being launched with the instance profile created in step 1.

    For more information about using the EC2 CLI, see ec2-run-instances and ec2-describe-instances in the *Amazon Elastic Compute Cloud Command Line Reference*.

    ```
    $ ec2-run-instances -t t1.micro -p
         arn:aws:iam::111111111111:instance-profile/s3access -k key-pair -g
    'Web Server' ami-e565ba8c
                    RESERVATION      r-11c62773       111111111111     sg-e7ddc68e

                    INSTANCE         i-9a6843fd       ami-e565ba8c
         pending key-pair     0       us-east-1e          aki-88aa75e1
         disabled

    $ ec2-describe-instances
        RESERVATION      r-11c62773       111111111111     sg-e7ddc68e
                    INSTANCE         i-9a6843fd       ami-e565ba8c
                    ec2-50-19-200-155.compute-1.amazonaws.com        ip-10-28-
    28-186.ec2.internal
                    running key-pair     0       t1.micro        2012-04-
    26T16:29:25.000Z    us-east-1e
                    aki-88aa75e1            disabled

                         AIPAJ6OQOSP4IRHXCI6E4
    ```

3. Sign service requests with the AWS security credentials made available on the EC2 instance. The security credentials that are made available from instance metadata are temporary security credentials.

For information on using temporary security credentials, see Using Temporary Security Credentials (p. 361). This is handled for you if you develop your application with the AWS SDK.

If you are not using the AWS SDK, then you need to programmatically retrieve the AWS security credentials to sign your service API requests. The following example shows AWS security credentials returned for an IAM role named **s3access**.

```
GET http://169.254.169.254/latest/meta-data/iam/security-credentials/s3access;
 echo {
    "Code" : "Success",
    "LastUpdated" : "2012-04-26T16:39:16Z",
    "Type" : "AWS-HMAC",
    "AccessKeyId" : "AKIAIOSFODNN7EXAMPLE...",
    "SecretAccessKey" : "UtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY...",
    "Token" : "...",
    "Expiration" : "2012-04-26T22:39:16Z"
  }
```

Elastic Network Interfaces

Topics

Each EC2 instance has a default network interface that is assigned a primary private IP address on your Amazon VPC network. You can create and attach additional network interfaces, known as an elastic network interface (ENI), to any Amazon EC2 instance in your VPC. The maximum number of elastic network interfaces varies by instance type. For more information, see Private IP Addresses Per ENI Per Instance Type (p. 88).

ENIs have several attributes including, a primary private IP address, one or more secondary private IP addresses, an elastic IP address (optional), a MAC address, membership in specified security groups, a description, and a source/destination check flag. You can create an elastic network interface, attach it to an instance, detach it from an instance, and attach it to another instance. An ENI's attributes, including the private IP address, any secondary private IP addresses, elastic IP addresses, and MAC address, will follow the ENI as it is attached or detached from an instance and reattached to another instance.

You can attach an ENI to an instance during the launch process (cold attach), when an instance is stopped (warm attach), and when an instance is running (hot attach).

Attaching more than one network interface to an instance is useful when you want to:

- Create a management network.
- Use network and security appliances in your VPC.
- Create dual-homed instances with workloads/roles on distinct subnets.
- Create a low budget high availability solution.

Creating a Management Network

You can create a management network by utilizing ENIs. In a management network scenario the secondary network interface on the instance handles public-facing traffic and the primary network interface handles back-end management traffic and is connected to a separate subnet in your VPC that has more restrictive

access controls. The public facing interface, which may or may not be behind a load balancer, will have an associated security group allowing access to the server from the Internet (e.g.; allow TCP port 80 and 443 from 0.0.0.0/0, or from the load balancer) while the private facing interface will have an associated security group allowing SSH access only from an allowed range of IP addresses either within the VPC or from the Internet, a private subnet within the VPC or a virtual private gateway.

To ensure failover capabilities, consider using a secondary private IP for incoming traffic on a network interface. In the event of an instance failure, you can move the interface and/or secondary private IP address to a standby instance.
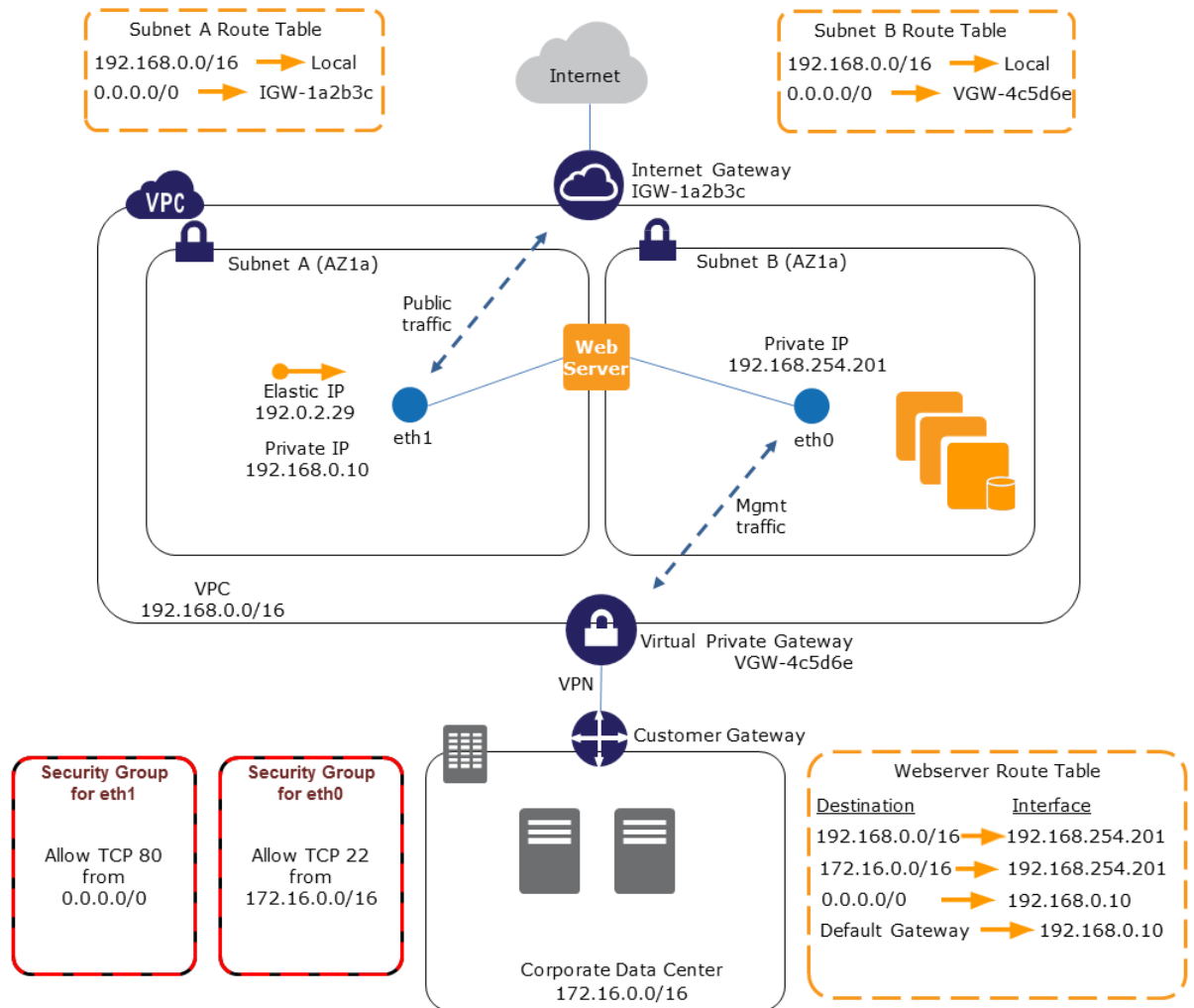
Use Network and Security Appliances in Your VPC

Some third party network and security appliances such as load balancers, network address translation (NAT) servers, and proxy servers prefer to be configured with multiple network interfaces. You can create and attach secondary network interfaces to instances in a VPC that are running these types of applications and configure the additional interfaces with their own public and private IP addresses, security groups, and source/destination checking.

Creating Dual-homed Instances with Workloads/Roles on Distinct Subnets

You can place an ENI on each of your web servers that connects to a mid-tier network where an application server resides. The application server can also be dual-homed to a backend network (subnet) where the database server resides. Instead of routing network packets through the dual-homed instances, each dual-homed instance receives and processes requests on the front end, initiates a connection to the backend, and then sends requests to the servers on the backend network.

Create a Low Budget High Availability Solution

If one of your instances serving a particular function fails, its network interface can be attached to a replacement or hot standby instance pre-configured for the same role in order to rapidly recover the service. For example, you can use an ENI as your primary or secondary network interface to a critical service such as a database instance or a NAT instance. If the instance fails, you (or more likely, the code running on your behalf) can attach the ENI to a hot standby instance. Because the interface maintains its private IP addresses, Elastic IP addresses, and MAC address, network traffic will begin flowing to the standby instance as soon as you attach the ENI to the replacement instance. Users will experience a brief loss of connectivity between the time the instance fails and the time that the ENI is attached to the standby instance, but no changes to the VPC route table or your DNS server are required.

Best Practices for Configuring Network Interfaces
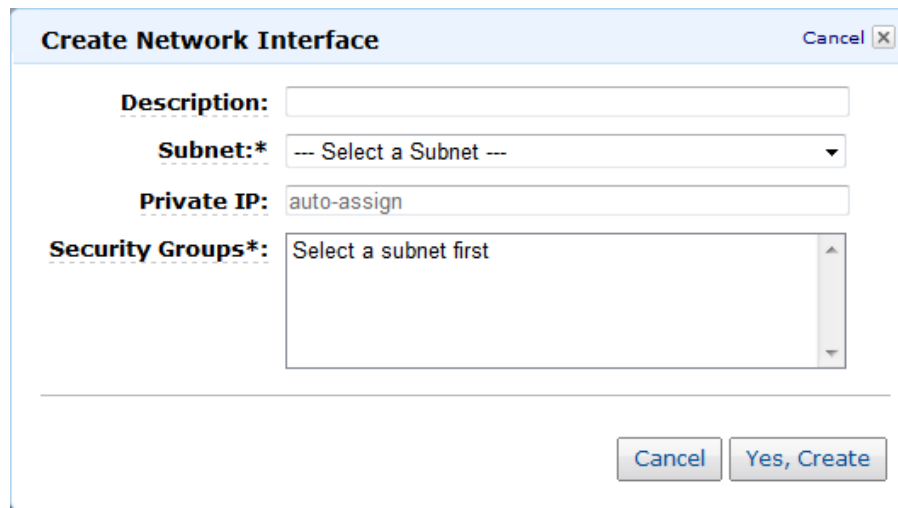
- You can attach an ENI to an instance when it's running (hot attach), when it's stopped (warm attach), or when the instance is being launched (cold attach).
- You can detach secondary (eth-n) network interfaces when the instance is running or stopped. However, you cannot detach the primary (eth0) interface.
- You can attach an ENI in one subnet to an instance in another subnet in the same VPC, however, both the network interface and the instance must reside in the same Availability Zone.
- When launching an instance from the CLI or API you can specify the ENIs to attach to the instance for both the primary (eth0) and additional network interfaces.
- Launching an EC2 instance with multiple network interfaces will automatically configure interfaces, private IP addresses and route tables on the operating system of the instance. Warm or hot attaching an additional network interface (when the instance is stopped or running) may require you to manually bring up the second interface, configure the private IP address, and modify the route table accordingly. Microsoft Windows Server 2003 and 2008 instances will automatically recognize the warm or hot attach and will automatically configure themselves. Instances that are not running a Microsoft Windows Server operating system might require the manual interaction to configure the interface on the operating system.)
- Attaching multiple network interfaces to an instance is intended to assist with the above use cases. Attaching multiple network interfaces to a single instance is not recommended as a method to provide multiple public IP addresses on a single instance.
- Attaching another network interface to an instance is not a method to increase or double the network bandwidth to or from the dual-homed instance.

Create an Elastic Network Interface

To create an ENI

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the **Navigation** pane, click **Network Interfaces**.

3. At the top of the **Network Interfaces** pane, click **Create Network Interface**.
4. In the **Create Network Interface** dialog box, in the **Description** box, type a descriptive name for the network interface.
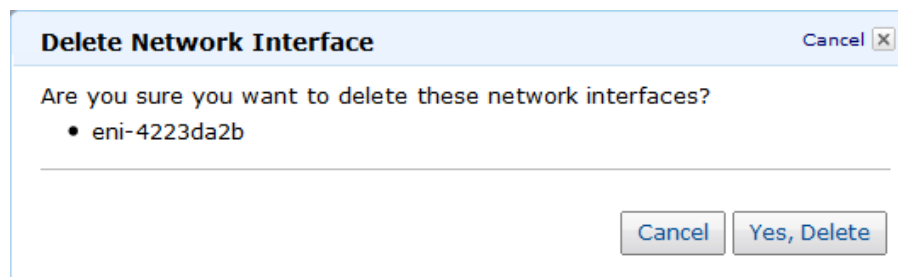
5. In the **Subnet** drop-down list box, select the subnet in which you want to create the network interface. The network interface cannot be moved to another subnet after it is created.
6. In the **Private IP** box, enter the primary private IP address for the network interface. If you don't specify an IP address, AWS will automatically assign one to the interface from an available IP address within the selected subnet.
7. In the **Security Groups** drop-down list box, select one or more security groups to use with the network interface.
8. Click **Yes, Create**.

Delete an Elastic Network Interface

You must first detach an ENI from an instance before you can delete it. Deleting an ENI releases all attributes associated with the network interface and releases any private IP addresses or elastic IP addresses for use by another instance.

To delete an ENI

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the **Navigation** pane, click **Network Interfaces**.
3. At the top of the **Network Interfaces** pane, select an ENI, and then click **Delete Network Interface**.
4. In the **Delete Network Interface** dialog box, click **Yes, Delete**.

Viewing Details about an Elastic Network Interface

You can view the attributes of your ENIs.

To view details about an ENI

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the **Navigation** pane, click **Network Interfaces**.
3. In the **Network Interfaces** pane, select an ENI.
4. In the lower pane, on the **Details** tab, you can view the details about the ENI.

Attach an Elastic Network Interface When Launching an Instance

To attach an ENI when launching an instance

You can attach an additional network interface, designated as eth1-*n*, to an instance when you launch it into a VPC.

Note

If an error occurs when attaching an ENI to your instance, this will cause the instance launch to fail.

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Click **Launch Instance** and select **Classic Wizard**.

3. In the **Request Instances Wizard**, choose an Amazon Machine Image (AMI).

4. On the **INSTANCE DETAILS** page, set the number and size of instance you want to launch, and then click the **Launch Instances** radio button.

5. On the **EC2** tab, set the availability zone, and then click the **VPC** tab and confirm that your subnet is selected in the **Subnet** list box.

6. Click **Continue**.

7. On the **Advanced Instance Options** page, enter the number of network interfaces that you want to attach to the instance.

   You can enter an IP address for the primary network interface (eth0) on this page.

8.   Click **Continue**, and then on the **CREATE KEY PAIR** page, select an existing key pair or create a new one. If you create a new key pair, you must download it before you can click **Continue**.

Note

A *key pair* is a security credential similar to a password, which you use to securely connect to your instance once it's running. If you're new to Amazon EC2 and haven't created any key pairs, when the wizard displays the **Create Key Pair** page, the **Create a new Key Pair** button is selected by default. You'll need to create a key pair. For more information, see Getting an SSH Key Pair (p. 255).

When you create a new key pair, EC2 uses the name you specify (e.g., `mykeypair`) to name the private key file (with a `.pem` extension) associated with the key pair.

9. On the **CONFIGURE FIREWALL** page, select an existing security group for the primary network interface or create a new one, and then click **Continue**.

The security group for the additional network interface was previously selected when the ENI was created.

Request Instances Wizard

CHOOSE AN AMI   INSTANCE DETAILS   CREATE KEY PAIR   **CONFIGURE FIREWALL**   REVIEW

Security groups determine whether a network port is open or blocked on your instances. You ma
or we can help you create a new security group to allow access to your instances using the sug
additional ports now or update your security group anytime using the Security Groups page.

Primary Network Interface

○ **Choose one or more of your existing Security Groups**

◉ **Create a new Security Group**

Group Name	quick-start-1
Group Description	quick-start-1

Inbound Rules

Create a new rule: Custom TCP rule ▼

Port range:
(e.g., 80 or 49152-65535)

Source: 0.0.0.0/0
(e.g., 192.168.2.0/24, sg-47ad482e, or
1234567890/default)

➕ Add Rule

TCP

Port (Service)   **Sourc**

22 (SSH)   0.0.0.0

‹ Back   **Continue** ▶

10. On the **REVIEW** page, details about the primary and additional network interface are displayed.
Review the settings, and then click **Launch**.

Request Instances Wizard

CHOOSE AN AMI   INSTANCE DETAILS   CREATE KEY PAIR   CONFIGURE FIREWALL   REVIEW

Please review the information below, then click **Launch**.

AMI: Amazon Linux AMI ID ami-3c0b4a6e (x86_64)

Name: Amazon Linux AMI 2012.03

Description: The Amazon Linux AMI 2012.03 is an EBS-backed, PV-GRUB image.
includes Linux 3.2, AWS tools, and repository access to multiple
versions of MySQL, PostgreSQL, Python, Ruby, and Tomcat.

Number of Instances:	1
VPC ID:	vpc-f28a359b
VPC Subnet:	subnet-cd8a35a4 (10.0.1.0/24)
Availability Zone:	No Preference
Instance Type:	High-CPU Medium (c1.medium)
Instance Class:	On Demand

Monitoring:	Disabled	**Termination Protection:**	Disabled
Tenancy:	Default		
Kernel ID:	Use Default	**Shutdown Behavior:**	Stop
RAM Disk ID:	Use Default		
Network Interfaces:	1		
Primary IP Addresses:	1 auto-assigned		
Secondary IP Addresses:	1 auto-assigned		
User Data:			

‹ Back   **Launch** ▶

11. On the confirmation page, click **Close**, and then in the **Navigation** pane, click **Instances** to view
your instance's status. It takes a short time for an instance to launch. The instance's status is *pending*
while it's launching. After a short period, your instance's status switches to *running*. You can click
Refresh to refresh the display.

Attach an Elastic Network Interface to a Stopped or Running Instance

You can attach an ENI to any of your stopped or running instances in your Amazon VPC from either the Instances page or the Network Interfaces page of the EC2 console.
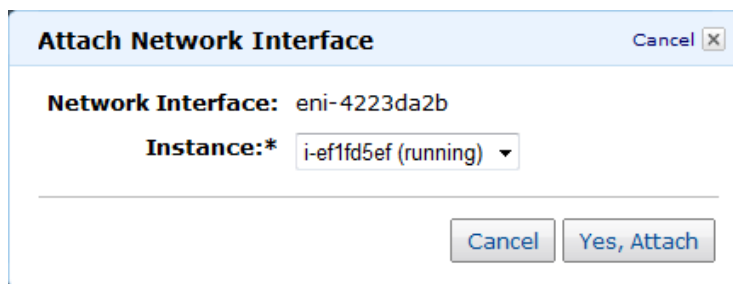
To attach an ENI to a stopped or running instance on the Instances page
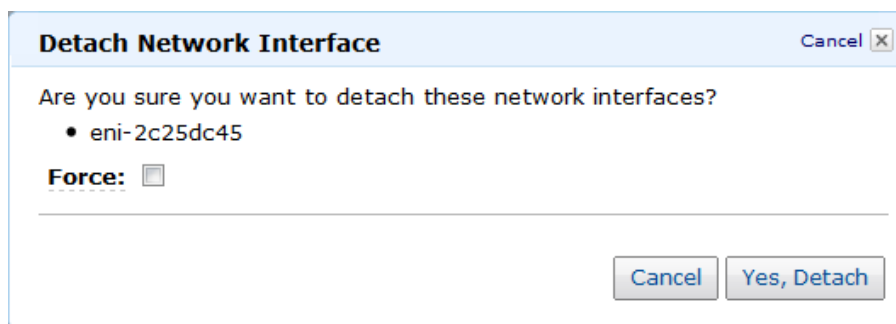
1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the **Navigation** pane, click **Instances**.
3. In the **My Instances** pane, right-click an EC2 instance, and then click **Attach Network Interface**.
4. In the **Attach Network Interface** dialog box, select the network interface you want to attach to the instance, and then click **Yes, Attach**.

To attach an ENI to a stopped or running instance on the Network Interfaces page

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the **Navigation** pane, click **Network Interfaces**.
3. In the **Networking Interfaces** pane, select an ENI.

4. At the top of the **Network Interfaces** pane, click **Attach**.

5. In the **Attach Network Interface** dialog box, in the **Instances** pop-up list box, select the instance you want to attach the ENI to, and then click **Yes, Attach**.

Attach Network Interface   Cancel ☒

Network Interface: eni-4223da2b

Instance:*   [i-ef1fd5ef (running) ▾]

[Cancel] [Yes, Attach]

Detach an Elastic Network Interface from an Instance

You can detach an eth1 elastic network interface (ENI) any time from either the Instances page or the Network Interfaces page of the EC2 console.

To detach an ENI from an instance on the Instances page

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

2. In the **Navigation** pane, click **Instances**.

3. In the **My Instances** pane, right-click an EC2 instance, and then click **Detach Network Interface**.

4. In the **Detach Network Interface** dialog box, select the network interface you want to detach from the instance, and then click **Yes, Detach**.

To detach an ENI from an instance on the Network Interfaces page

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

2. In the **Navigation** pane, click **Network Interfaces**.

3. In the **Network Interfaces** pane, select an ENI, and then click **Detach**.

4. In the **Detach Network Interface** dialog box, click **Yes, Detach**.

   Note

   In some cases the ENI may fail to detach from the instance. If this occurs, select the **Force** checkbox, and then try again.

Detach Network Interface   Cancel ☒

Are you sure you want to detach these network interfaces?
   • eni-2c25dc45

Force: ☐

[Cancel] [Yes, Detach]

Change an Elastic Network Interface's (ENI's) Security Group

You can change the security groups that are associated with elastic network interfaces (ENIs).

Note

Security group membership for interfaces owned by other Amazon Web Services, such as Elastic Load Balancing, cannot be changed in the **Network Interfaces** page or by using the EC2 API\CLI. To modify the security group membership of an interface owned by one of these services, use the service specific console or API\CLI commands.

To change an ENI's security group

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the **Navigation** pane, click **Network Interfaces**.
3. In the **Network Interfaces** pane, select an ENI.
4. At the top of the **Network Interfaces** pane, click **Change Security Groups**.
5. In the **Change Security Groups** dialog box, in the **Security Groups** list box, select the security groups you want to use, and then click **Save**.
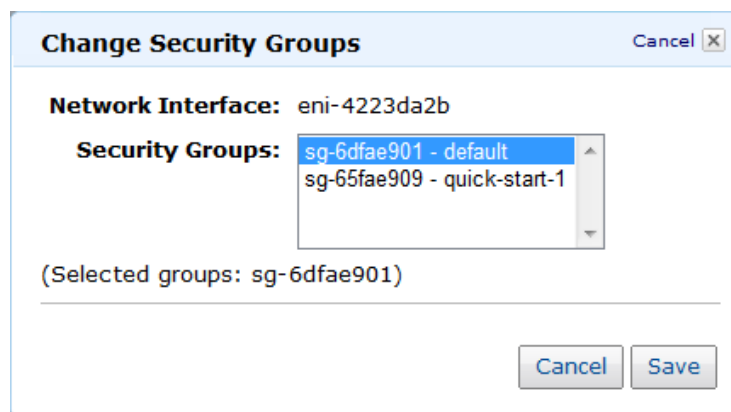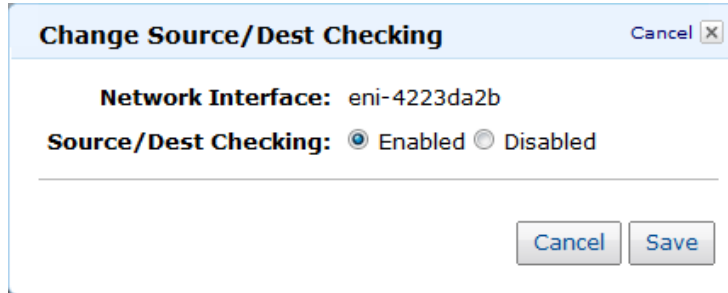
Change Source/Destination Checking for an Elastic Network Interface

The Source/Destination Check attribute controls whether source/destination checking is enabled on the instance. Disabling this attribute enables an instance to handle network traffic that isn't specifically destined for the instance. For example, instances running services such as network address translation, routing, or firewalls should set this value to disabled. The default value is enabled.

To change source/destination checking for an ENI

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the **Navigation** pane, click **Network Interfaces**.
3. In the **Network Interfaces** pane, right-click an ENI, and then click **Change Source/Dest Check**.
4. In the **Change Source/Dest Checking** dialog box, select **Enabled** (if enabling), or **Disabled** (if disabling), and the click **Save**.
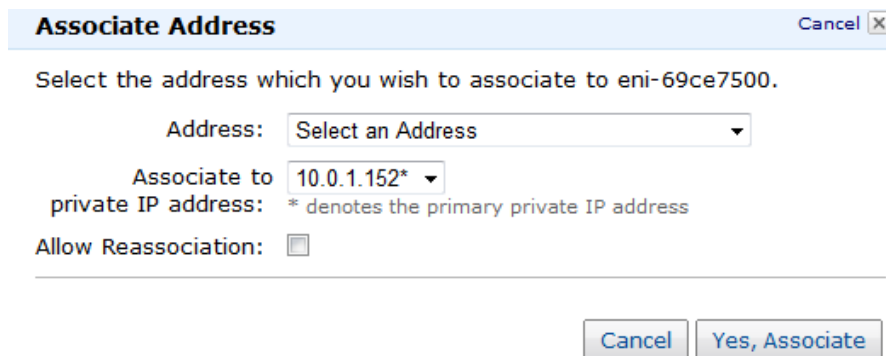
Associate an Elastic IP Address with an Elastic Network Interface

You can associate an elastic IP address with an elastic network interface (ENI).

To associate an elastic IP address with an ENI

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the **Navigation** pane, click **Network Interfaces**.
3. In the **Network Interfaces** pane, right-click a network interface, and then click **Associate Address**.
4. In the **Associate Address** dialog box, select the Elastic IP address you want to associate with your ENI.
5. In Associate to private address, select the private IP address to which you want to associate the Elastic IP address.
6. Click **Allow Reassociation** to allow the Elastic IP address to be associated with the specified network interface if it is currently associated with another instance or network interface
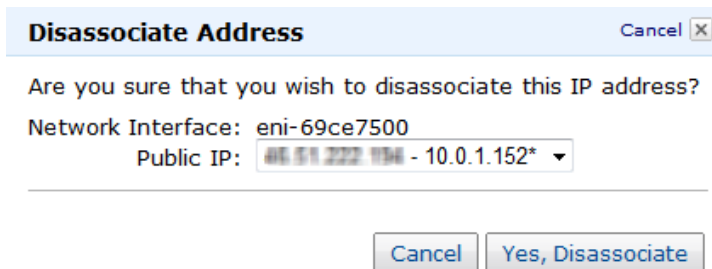7. Click **Yes, Associate**.

Disassociate an Elastic IP Address from an Elastic Network Interface

If the elastic network interface (ENI) has an elastic IP address associated with it, you can disassociate the address, and then either associate it with another ENI or release it back to the address pool.

To disassociate an elastic IP address from an ENI

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the **Navigation** pane, click **Network Interfaces**.
3. In the **Network Interfaces** pane, right-click a network interface, and then click **Disassociate Address**.
4. In the **Disassociate Address** dialog box, click **Yes, Disassociate**.

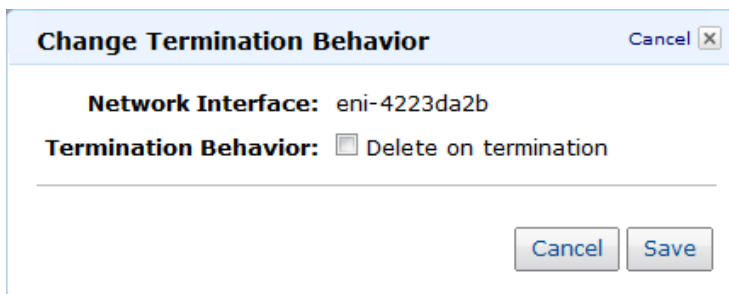# Change Termination Behavior for Elastic Network Interfaces

You can set the termination behavior for elastic network interfaces (ENIs) attached to an instance so that they are automatically deleted when you delete the instance they're attached to.

Note

By default, ENIs that are automatically created and attached to instances using the EC2 console (Create Network Interfaces button or the Launch Instances Wizard) are set to terminate when the attached instance terminates. However, ENIs created using the create-network-interface command are not set to be terminated when the instance is terminated.

To change termination behavior for ENIs

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the **Navigation** pane, click **Network Interfaces**.
3. In the **Network Interfaces** pane, right-click an ENI, and then click **Change Termination Behavior**.
4. In the **Change Termination Behavior** dialog box, select the **Delete on termination** check box if you want the ENI to be deleted when you terminate an instance.
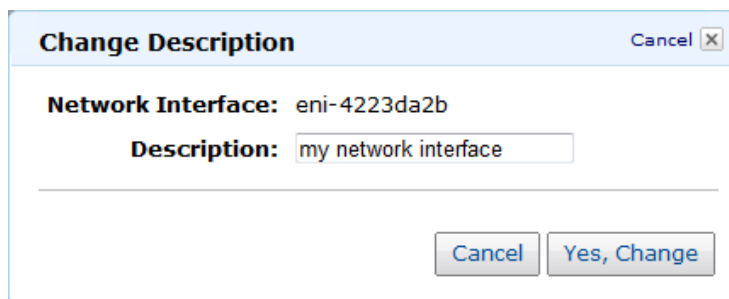
Add or Edit a Description for an Elastic Network Interface

To add or edit a description for an ENI

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the **Navigation** pane, click **Network Interfaces**.
3. In the **Network Interfaces** pane, right-click and ENI, and then click **Change Description**.
4. In the **Change Description** dialog box, enter a description for the ENI, and then click **Yes, Change**.

Add or Edit Tags for an Elastic Network Interface

Tags are metadata you can add to an elastic network interface (ENI). Each tag consists of a key and an optional value. For more information about using tags, see Tagging Your Resources (p. 508).

Tags are private and are only visible to your account. Tags which have keys that begin with aws: have been created by AWS and cannot be edited or deleted.

To add or edit tags for an ENI

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the **Navigation** pane, click **Network Interfaces**.
3. In the **Network Interfaces** pane, select an ENI.
4. In the lower pane, click the **Tags** tab, and then click **Add/Edit Tags**.
5. In the **Tag Network Interfaces** dialog box, enter a key and an optional value for each tag that you want to add, and then click **Save Tags**.

Tag Network Interfaces

Cancel ✕

Add tags to your network intefaces to simplify the administration of your EC2 infrastructure. A form of metadata, tags consist of a case-sensitive key/value pair, are stored in the cloud and are private to your account. You can create user-friendly names that help you organize, search, and browse your resources. For example, you could define a tag with key = Name and value = CRMwebroot. You can add up to 10 unique keys to each volume along with an optional value for each key. For more information, go to Using Tags in the *EC2 User Guide*.

Key (127 characters maximum)	**Value** (255 characters maximum)	**Remove**
Name		✖
		✖

Add another Tag. (Maximum of 10)

Cancel   Save Tags

Instance IP Addresses

Topics

This section describes the types of IP addresses available to Amazon EC2 instances, including Elastic IP addresses, which you can remap on demand.

Public and Private Addresses

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

Starting with API version 2012-06-15, you can choose to add one or more secondary private IP addresses to an instance launched in Amazon Virtual Private Cloud (Amazon VPC) and assign these to the network interface. For more information, see Multiple IP Addresses (p. 386).

Amazon EC2 also provides an internal DNS name and a public DNS name that 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.

Note

If you require persistent Internet routable IP addresses that can be assigned to and removed from instances as necessary, use Elastic IP addresses. For more information, see Elastic IP Address Concepts (p. 402).

Private (RFC 1918) Addresses and Internal DNS

All Amazon EC2 instances are allocated a private IP address by DHCP. These ranges are defined in RFC 1918, are only routable within Amazon EC2, and are used for communication between instances. For more information, go to RFC 1918.

Note

For instances launched in VPC, you can assign one or more secondary private IP addresses to a network interface on the instance. For more information, see Multiple IP Addresses (p. 386).

In Amazon EC2, this private IP address is associated exclusively with the instance for its lifetime and is only returned to Amazon EC2 when the instance is stopped or terminated. In Amazon VPC, an instance retains its private IP addresses when the instance is stopped.

Each instance is provided an internal DNS name that resolves to the private IP address of the instance from within Amazon EC2; it will not resolve outside of Amazon EC2.

Public Addresses and DNS

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

This public address is associated exclusively with the instance until it is stopped, terminated or replaced with an Elastic IP address.

> **Important**
>
> Amazon EC2 instances that access other instances through their public NAT IP address are charged for Regional or Internet data transfer, depending on whether the instances are in the same Region.

Each instance is provided an external DNS name that resolves to the public IP address of the instance outside the Amazon EC2 network and the private IP address from within Amazon EC2 network.

IP Addresses for Instances in a VPC

When you launch an instance in a VPC, you can optionally specify a primary IP address for the instance. If you don't specify a primary private IP address, an IP address in the subnet's range is automatically assigned. The assigned address stays with the instance until the instance is terminated. Even if you stop and restart the instance, it retains the same primary private IP address.

You can assign additional IP addresses, known as secondary private IP addresses, to Amazon EC2 instances that are running in Amazon VPC. Unlike primary private IP addresses, secondary private IP addresses can be reassigned from one network interface to another and from one instance to another. For more information about multiple IP addresses, see Multiple IP Addresses (p. 386).

For more information about Amazon VPC, go to the Amazon Virtual Private Cloud Getting Started Guide.

Determining Your IP Addresses

This section describes how to determine the internal and external IP addresses of an instance you own.

AWS Management Console

To determine your instance's public and private IP addresses

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Click **Instances** in the **Navigation** pane.
   The console displays a list of running instances.
3. Locate and select an instance.
   The console displays information about the instance in the lower pane.
4. To determine the public IP address, use the IP address specified within the **Public DNS** field.
5. To determine the private IP address, look at the **Private IP Address** field.

Command Line Tools

To determine your instance's public and private IP addresses

- Use the **ec2-describe-instances** command with the instance ID. The instance must be running.

```
PROMPT>  ec2-describe-instances instance_id
RESERVATION   r-f25e6f9a 111122223333  default
INSTANCE <instance_id> ami-b232d0db ec2-204-236-202-134.compute-1.amazon
aws.com domU-12-31-39-00-86-35.compute-1.internal running gsg-keypair 0
m1.small 2010-03-30T08:43:48+0000 us-east-1a aki-94c527fd ari-96c527ff mon
itoring-disabled 204.236.202.134 10.254.137.191 ebs
BLOCKDEVICE     /dev/sda1       vol-cf13b3a6    2010-03-30T08:01:44.000Z
```

In the preceding example, the public IP address is 204.236.202.134. The private IP address is
10.254.137.191.

API

To determine your instance's public and private IP addresses

1. Construct the following Query request.

```
https://ec2.amazonaws.com/
?Action=DescribeInstances
&InstanceId.1=instance-id
&...auth parameters...
```

Following is an example response.

```
<DescribeInstancesResponse xmlns="http://ec2.amazonaws.com/doc/2012-07-20/">

  <reservationSet>
    <item>
      <reservationId>r-44a5402d</reservationId>
      <ownerId>999988887777</ownerId>
      <groupSet>
        <item>
          <groupId>default</groupId>
        </item>
      </groupSet>
      <instancesSet>
        <item>
          <instanceId>i-28a64341</instanceId>
          <imageId>ami-6ea54007</imageId>
          <instanceState>
             <code>16</code>
             <name>running</name>
          </instanceState>
           <privateDnsName>ip-10-203-46-149.ec2.internal</privateDnsName>
           <dnsName>ec2-75-101-217-237.compute-1.amazonaws.com</dnsName>
           ...
           <privateIpAddress>10.203.46.149</privateIpAddress>
           <ipAddress>75.101.217.237</ipAddress>
           ...
        </item>
      </instancesSet>
    </item>
  </reservationSet>
</DescribeInstancesResponse>
```

2. To determine the public IP address, use the IP address specified within `ipAddress`.

3. To determine the private IP address, use the IP address specified within `privateIpAddress`.

From Within the Instance

To determine your private IP address in Linux/UNIX

1. Connect to the instance.

2. Enter one of the following commands:

   - `#`   `ifconfig eth0`

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

     The second option refers to the instance metadata. For more information, see Instance Metadata (p. 302).

To determine your private IP address in Windows

1. Connect to the instance.

2. On the taskbar, click **Start**, right-click **My Computer**, and select **Properties**.

3. Click the **Computer Name** tab. The IP address appears in the **Full computer name** field.

To determine your public IP address

1. Connect to the instance.

2. Determine your public IP address from your instance by referring to the instance data.
   `PROMPT>` **`GET http://169.254.169.254/latest/meta-data/public-ipv4`**

Related Topics

- IP Information FAQ (p. 546)

Multiple IP Addresses

In Amazon Virtual Private Cloud (Amazon VPC) each EC2 instance has a default network interface that is assigned an IP address on your Amazon VPC network. When you launch an instance in a VPC, you can optionally specify an IP address for the instance. If you don't specify a primary private IP address, a primary IP address in the subnet's range is automatically assigned. The assigned address stays with the instance until the instance is terminated. Even if you stop and restart the instance, it retains the same primary IP address.

Beginning with API version 2012-06-15, you can assign additional IP addresses, known as secondary private IP addresses, to EC2 instances that are running in Amazon VPC. Unlike primary private IP addresses, secondary private IP addresses can be reassigned from one network interface to another and from one instance to another.

You can associate an Elastic IP address with any primary or secondary private IP address so that the instance can accept external traffic on the ports you specify.

Note

The number of secondary private IP addresses that you can assign varies by instance type. For more information, see Instance Families and Types (p. 85).

Use Cases

Assigning multiple private IP addresses to an EC2 instance in your VPC is useful when you want to do the following:

- Host multiple websites on a single server by doing either of the following:
  - Using multiple SSL certificates on a single server and associating each certificate with a specific IP address.
  - Configuring multiple virtual hosts on a web server.
- Operate network appliances, such as firewalls or load balancers, that have multiple IP addresses for each network interface.
- Redirect internal traffic to a standby EC2 instance in case your instance fails, by reassigning the secondary private IP address to the EC2 instance.

Requirements for Multiple IP Addresses

The following list explains how multiple IP addresses work with other components. It also describes the requirements for managing multiple IP addresses.

- You can assign secondary private IP addresses to any elastic network interface (eth0 to ethn).
- Secondary private IP addresses must belong to the subnet CIDR block in which the elastic network interface exists.
- Security groups apply to interfaces, not to IP addresses. IP addresses are subject to the security group of the interface to which they're assigned.
- Secondary private IP addresses can be assigned and unassigned to elastic network interfaces that are attached or unattached to instances.
- Secondary private IP addresses can be assigned and unassigned to elastic network interfaces attached to running or stopped instances.
- Secondary private IP addresses that are assigned to one interface can be reassigned to another elastic network interface if you explicitly allow this using the console, the command line tools, or the API.
- Each private IP address can only be associated with a single Elastic IP address, and vice versa.
- When a secondary private IP address is reassigned to another interface, the secondary private IP address retains its association with an Elastic IP address.
- When a secondary private IP address is unassigned from an interface, the associated Elastic IP address (if it exists) is automatically disassociated from the secondary private IP address.
- When assigning multiple secondary private IP addresses to a network interface using command line tools or API, the entire operation will fail if one of the secondary private IP addresses cannot be assigned.
- Primary private IP addresses, secondary private IP addresses and any associated Elastic IP addresses remain with the interface when the interface is detached from an instance and/or attached to another instance.
- Although you cannot move the primary network interface from an instance, you can reassign the secondary private IP address of the primary network interface to another network interface.
- You can move any additional network interface from one instance to another.

Assign a Secondary Private IP Address

You can assign a secondary private IP address:

- When you launch an instance in a VPC.
- When you create an elastic network interface in a VPC using the command line or API.
- When an instance is running or stopped.
- When an elastic network interface is attached or unattached to an instance.

Assigning a secondary private IP address to an instance involves the following tasks:

Step 1: Assign a Secondary Private IP Address

To assign a secondary private IP address when launching an instance

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Click **Launch Instance**.
3. On the **Create a New Instance** page, click **Classic Wizard**, and then click **Continue**.
4. Click **Launch Instance** and select **Classic Wizard**.

5. In the **Request Instances Wizard**, choose an Amazon Machine Image (AMI).

6. On the **INSTANCE DETAILS** page, set the number and size of the instance you want to launch, and then click the **Launch Instances** radio button.

7. On the **EC2** tab, set the Availability Zone, and then click the **VPC** tab and confirm that your subnet is selected in the **Subnet** list box.

8. Click **Continue**.

9. Under **Advanced Instance Options**, in **Number of Network Interfaces**, click the number of network interfaces that you want to attach to the instance. The number of network interfaces that you can attach varies by instance type. For more information, see Instance Families and Types (p. 85).

    For each network interface, you can manually specify a primary private IP address and one or more secondary private IP addresses. For this example, however, we'll accept the IP address that is automatically assigned.

10. Beside **Secondary IP Addresses**, click **Add**.

11. In the **Address** field, type a private IP address that you want to assign to the interface, or accept the default, auto-assign, to let AWS automatically assign an address. The address that you enter must be within the subnet range for the EC2 instance.

Important

After you have added a secondary private IP address to an instance, you must connect to
the instance and configure the secondary private IP address on the instance itself. For more
information, see Step 2: Configure the Operating System on Your Instance to Recognize
the Secondary Private IP Address (p. 394) .

Request Instances Wizard

CHOOSE AN AMI   INSTANCE DETAILS   CREATE KEY PAIR   CONFIGURE FIREWALL   REVIEW

Number of Instances: 1   **Availability Zone:** No Preferenc

Advanced Instance Options

Here you can choose a specific kernel or RAM disk to use with your instances. You can also choos
Detailed Monitoring or enter data that will be available from your instances once they launch.

Kernel ID: Use Default   ▼   **RAM Disk ID:**   Use Default

Monitoring: ☐ Enable CloudWatch detailed monitoring for this instance
(additional charges will apply)

User Data:

⦿ **as text**

◯ **as file**   ☐ base64 encoded

Termination   ☐ Prevention against accidental termination.   **Shutdown**   Stop
Protection: **Behavior:**

IAM Role: ❓   None ▼   **Tenancy:**   Default

Number of Network Interfaces: 1 ▼

eth0	**Network Interface:**	New Interface	▼	**Secondary IP**	auto-assign
	Subnet:	subnet-cd8a35a4 (10.0.1.0/24)	▼	**Addresses:** Add	
	IP Address:	auto-assign			

‹ Back   **Continue** ▶

12. Click **Continue**.
13. (Optional) Give your instance a user friendly name or "tag," and then click **Continue**.
14. On the **Create Key Pair** page, do one of the following:

   • Click **Choose from your existing Key Pairs**. In the **Your existing Key Pairs** box, click the key
     pair that you want to associate with this instance.
   • Click Create a new Key Pair, and then follow the instructions on the screen. Save the key pair file
     that is created to a place where you can find it again when you need it later.

- (Not recommended) Click Proceed without a Key Pair. Without a key pair, you will need the user name and password that are associated with the AMI used to launch the instance.

    For more information, see Getting an SSH Key Pair (p. 255).

15. On the **CONFIGURE FIREWALL** page, select an existing security group for the primary network interface or create a new one, and then click **Continue**.

    The security group for the additional network interface was previously selected when the ENI was created.

16. On the **REVIEW** page, details about the instance including the secondary private IP addresses are displayed. Review the settings, and then click **Launch**.

Request Instances Wizard

✔	✔	✔	✔	◯—
CHOOSE AN AMI	INSTANCE DETAILS	CREATE KEY PAIR	CONFIGURE FIREWALL	REVIEW

Please review the information below, then click **Launch**.

AMI: Amazon Linux AMI ID ami-3c0b4a6e (x86_64)
Name: Amazon Linux AMI 2012.03
Description: The Amazon Linux AMI 2012.03 is an EBS-backed, PV-GRUB image. includes Linux 3.2, AWS tools, and repository access to multiple versions of MySQL, PostgreSQL, Python, Ruby, and Tomcat.

Number of Instances: 1
VPC ID: vpc-f28a359b
VPC Subnet: subnet-cd8a35a4 (10.0.1.0/24)
Availability Zone: No Preference
Instance Type: High-CPU Medium (c1.medium)
Instance Class: On Demand

Monitoring: Disabled   **Termination Protection:** Disabled
Tenancy: Default
Kernel ID: Use Default   **Shutdown Behavior:** Stop
RAM Disk ID: Use Default
Network Interfaces: 1
Primary IP Addresses: 1 auto-assigned
Secondary IP Addresses: 1 auto-assigned
User Data:

‹ Back   Launch ▶

17. On the confirmation page, click **Close**, and then in the **Navigation** pane, click **Instances** to view your instance's status. It takes a short time for an instance to launch. The instance's status is *pending*

while it's launching. After a short period, your instance's status switches to *running*. You can click
Refresh to refresh the display.

> **Note**
>
> To display the VPC ID, subnet ID, and private IP address of the instance, you might need
> to click **Show/Hide**. In the list of instance attributes, select the check boxes corresponding
> to the attributes that you want to view, and clear the others.

As noted earlier, after the instance is launched, you must configure the local operating system to recognize
any secondary private IP addresses. For more information, see Step 2: Configure the Operating System
on Your Instance to Recognize the Secondary Private IP Address (p. 394)

To assign a secondary private IP to an existing instance

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the Navigation pane, click **Instances**.
3. Right-click an instance, and then click **Manage Private IP Addresses**.
4. In the **Manage Private IP Addresses** dialog box, click **Assign a secondary private address**.
5. In the **Address** box, do one of the following:

    *   Accept the default, auto-assign, and AWS will automatically assign a secondary private IP address
        within the subnet range for the instance.
    *   Enter a specific IP address that is within the subnet range for the instance.

6. To allow the secondary private IP address to be reassigned to the specified instance if it is already
    assigned to another instance or network interface, select the Allow reassignment check box.

7. Click **Yes, Update**, and then click **Close**.

To assign a secondary private IP address to an elastic network interface

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the Navigation pane, click **Network Interfaces**.
3. Right-click the network interface that you want to add a secondary private IP address to.
4. Click **Manage Private IP Addresses**.

5.   In the **Manage Private IP Addresses** dialog box, click **Assign a secondary private address**.

6.   In the Address box, do one of the following:

   • Accept the default, **auto-assign**, and AWS will automatically assign a secondary private IP address within the subnet range for the instance.
   • Enter a specific IP address that is within the subnet range for the instance.

7.   To allow the secondary private IP address to be reassigned to the specified elastic network interface if it is already assigned to another instance or network interface, select the **Allow reassignment** check box.
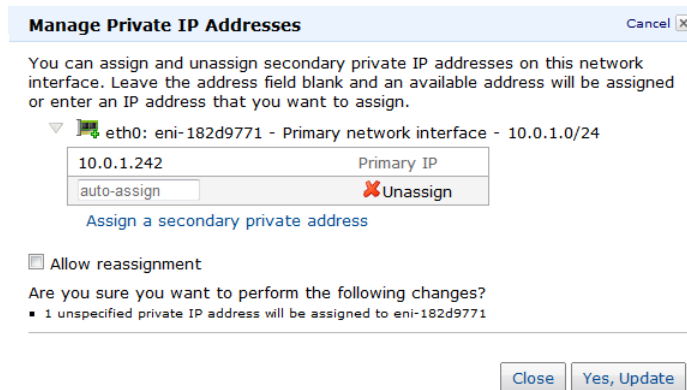
8.   Click **Yes, Update**, and then click **Close**.

Step 2: Configure the Operating System on Your Instance to Recognize the Secondary Private IP Address

After you have assigned a secondary private IP address to your instance, you need to configure the operating system on your instance to recognize the secondary private IP address. Use one of the following procedures:

Configure the Operating System on a Windows Instance

Configuring the operating system on a Windows Instances to recognize a secondary private IP address requires the following:

> **Important**
>
> As a best practice, launch your Windows instances using the latest AMIs. If you are using an
> older Windows AMI, ensure that it has the Microsoft hot fix referenced in
> http://support.microsoft.com/kb/2582281.

Step 2.1 Configure Static IP Addressing on Your Windows Instance

To use multiple IP address on a Windows instance you need to configure your Amazon EC2 instance to
use static IP addressing rather than a DHCP server.

> **Important**
>
> When you configure static IP addressing on your instance, the IP address must match exactly
> what you have assigned it as shown in the AWS console, CLI, or API. If you enter these IP
> addresses incorrectly the instance could become unreachable.

To configure static IP addressing on the operating system of a Windows instance

> **Note**
>
> The following instructions are based on a Windows 2008 R2 operating system. The steps may
> vary based on the operating system of the Windows instance.

1.   Connect to your EC2 instance.
2.   Click **Start**, and then click **Control Panel**.
3.   Click **Network and Internet**, and then click **Network and Sharing Center**.
4.   Click the network interface (Local Area Connection).
5.   Click **Properties**.
6.   Click **Internet Protocol Version 4 (TCP/IPv4)**, and then click **Properties**.
7.   In the **Properties** dialog box, click Use the following IP address.
8.   Click **Start**. In the Search box, type `cmd`, and then press **Enter**. This opens the command prompt
     window.
9.   At the command prompt, type ipconfig /all, and then press Enter.
10. Note the current IPv4 address, default gateway, and DNS server for the network interface.
11. In the Internet Protocol Version 4 (TCP/IPv4) Properties dialog box, under Use the following IP
     address, in the IP address box, type the IPv4 address shown in the Command Prompt window.
12. In the Subnet mask box, type the subnet mask shown in the Command Prompt window.
13. In the Default gateway box, type the IP address of the default gateway shown in the Command
     Prompt window.
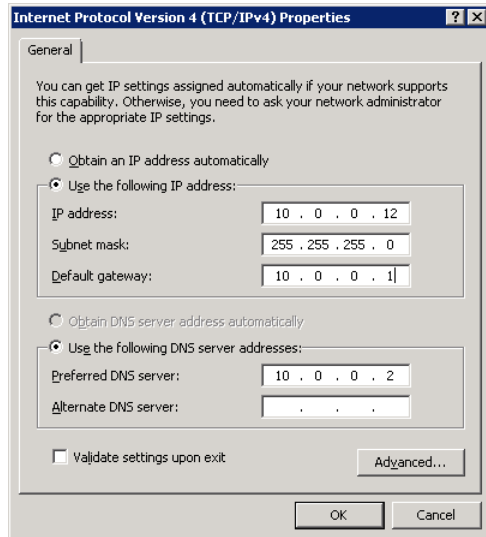
14. Click **OK**.

You will lose RDP connectivity to the Windows instance for a few seconds while the instance converts from using DHCP to static addressing. The instance will retain the same IP address information as before, but now this information is static and not managed by DHCP.

Step 2.2 Configure a Secondary Private IP Address on Your Windows Instance

If you want to add additional private IP addresses on this instance you'll need to follow the steps in the previous procedure to convert to static addressing before you can assign a secondary private IP address to the operating system.

To configure a secondary IP address on an instance

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the Navigation pane, click Instances.
3. Select your Amazon EC2 instance.
4. On the **Description** tab in the lower pane, note the secondary IP address.
5. Connect to your instance.
6. On your Windows instance, click **Start**, and then click **Control Panel**.
7. Click **Network and Internet**, and then click **Network and Sharing Center**.
8. Click the network interface (Local Area Connection).
9. Click **Properties**.
10. In the Local Area Connection Properties page, click **Internet Protocol Version 4 (TCP/IPv4)**, click **Properties**, and then click **Advanced**.
11. Click **Add**.
12. In the IP address box, type the secondary private IP address, and in the Subnet mask box, type the same subnet mask that you entered for the primary private IP address in Step 2.1 Configure Static IP Addressing on Your Windows Instance (p. 395).
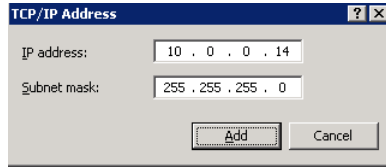
13. Click **Add**.
14. Click **OK**.
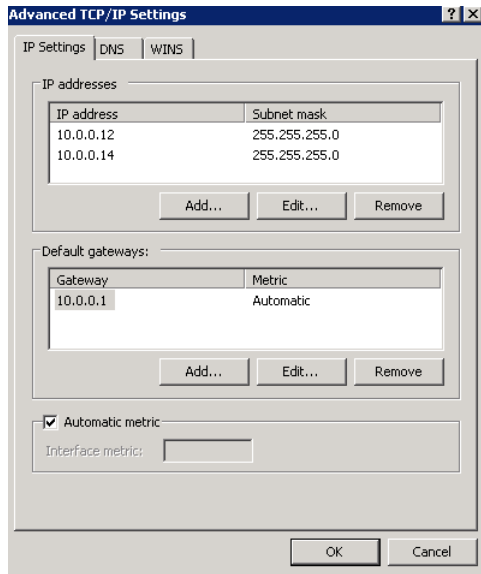
15. Click **OK** again, and then click **Close**.
16. To confirm that the secondary IP address has been added to the operating system, at a command prompt, type `ipconfig /all`, and then press **Enter**.

Step 2.3 Configure Any Applications to Use the Secondary Private IP Address

After the secondary private IP address has been added to the operating system, you must configure any application to use the secondary private IP address. For example, if your instance is running a website on IIS, you can configure a secondary private IP address that IIS can use.

To configure IIS to use the secondary private IP address

1. Connect to your EC2 instance.
2. Open Internet Information Services (IIS) Manager.
3. In the **Connections** pane, expand **Sites**.
4. Right-click your website, and then click **Edit Bindings**.
5. In the **Site Bindings** dialog box, under **Type**, click **http**, and then click **Edit**.
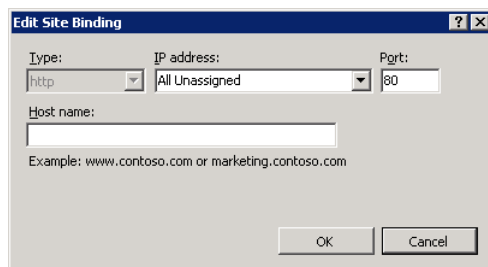
By default, each website accepts HTTP requests from all IP addresses

6.   In the **Edit Site Binding** dialog box, in the **IP address** box, click the secondary private IP address.

7.   Click **OK**, and then click **Close**.

8.   To allow Internet requests to your website, configure an Elastic IP address and associate it with the secondary private IP address. For more information, see

Configure the Operating System on a Linux Instance

Configuring the operating system on a Linux instance involves the following steps:

Step 2.1 Record Your Secondary Private IP Address

Before you configure your secondary private IP address, record the IP addresses configured on your instance using the AWS Management Console.

Note

You can also retrieve this information from the instance metadata service using a command similar to the following:

```
curl
http://169.254.169.254/latest/meta-data/network/interfaces/macs/<eni-mac-address
>/local-ipv4s
```

For more information, see Instance Metadata (p. 302).

To view the secondary private IP addresses configured on your Linux instance

1.   Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

2.   In the Navigation pane, click **Instances**.

3.   Select the check box corresponding to your Amazon EC2 instance.

4.   On the **Description** tab in the lower pane, note the secondary IP address.

Step 2.2 Retrieve the Current IP Address Information on the Instance

Type ifconfig to see the current IP addresses configured on your Linux operating system. You will see a response similar to the following. The second line shows the primary private IP address on the primary network interface (eth0). Note the subnet mask and use the same value when configuring secondary private IP addresses.

```
eth0      Link encap:Ethernet   HWaddr 02:78:D7:00:0D:D4
          inet addr:10.0.0.174  Bcast:10.0.0.255  Mask:255.255.255.0
          inet6 addr: fe80::78:d7ff:fe00:dd4/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:56930 errors:0 dropped:0 overruns:0 frame:0
          TX packets:60650 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:8137617 (7.7 MiB)  TX bytes:8787836 (8.3 MiB)
          Interrupt:32
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

Step 2.3: Configure a Secondary Private IP Address on the Linux Instance

The following procedure explains how to configure the operating system on an Amazon Linux instance to recognize a secondary private IP address. If you are using a different Linux instance type, refer to the directions for the specific operating system.

On Amazon Linux, the interface configuration files are stored in the `/etc/sysconfig/network-scripts/` directory. Each network interface configuration file is named `ifcfg-NAME` where `NAME` denotes the network interface name. The "ifcfg-eth0" file contains the interface configuration for the eth0 network interface.

To configure a secondary private IP address on the network interface

1. Connect to your Linux instance.
2. Using a text editor, create an appropriately named file in /etc/sysconfig/network-scripts/, for example `ifcfg-eth0:1`
3. Enter the configuration information for the secondary private IP address:

    ```
    DEVICE=eth0:1
    BOOTPROTO=static
    ONPARENT=yes
    IPADDR=<secondary-ip-address>
    NETMASK=<subnet-mask>
    ```

4. Save the file.
5. Run `sudo ifup eth0:1` to apply the updated configuration to the running system.

Step 3: Optionally Assign an Elastic IP Address to the Secondary Private IP Address

If you require a public IP address of the secondary private IP address, you can associate an Elastic IP address with secondary private IP address. For step-by-step instructions, see Using Elastic IP Addresses (p. 401).

Unassign a Secondary Private IP Address

If you no longer require a secondary private IP address, you can unassign it from the instance or the network interface. When a secondary private IP address is unassigned from an elastic network interface, the Elastic IP address (if it exists) is also disassociated.

To unassign a secondary private IP address from an instance

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the Navigation pane, click **Instances**.
3. Right-click an instance, and then click **Manage Private IP Addresses**.
4. In the **Manage Private IP Addresses** dialog box, beside the secondary private IP address that you want to unassign, click **Unassign**.

5. Click **Yes, Update.**, and then click **Close**.

To unassign a secondary private IP address from a network interface

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the Navigation pane, click **Network Interface**.
3. Right-click an instance, and the click **Manage Private IP Addresses**.
4. In the **Manage Private IP Addresses** dialog box, beside the secondary private IP address that you want to unassign, click **Unassign**.

5.   Click **Yes, Update**, and then click **Close**.

View Private IP Addresses

You can view the private IP addresses that are associated with an instance or a network interface from the Amazon EC2 console.

To view the private IP addresses assigned to an instance

1.   Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2.   In the Navigation pane, click **Instances**.
3.   Select the instance whose private IP addresses you want to view.
4.   In the details pane on the **Description** tab below the list of instances, you can view the primary private IP address and any secondary private IP addresses assigned to the instance.

Root Device Type:	ebs	**Tenancy:**	default
IAM Role:	-		
Lifecycle:	normal		
Block Devices:	sda1		
Network Interfaces:	eth0 eth1		
Public DNS:			
Private DNS:		**Product Codes:**	
Private IPs:	10.0.0.205, 10.0.0.133		
Secondary Private IPs:	10.0.0.13, 10.0.0.168		

Using Elastic IP Addresses

Topics

Elastic IP addresses are static IP addresses designed for dynamic cloud computing. An Elastic IP address is associated with your account, not a particular instance. You control addresses associated with your account until you choose to explicitly release them.

Important

If you're using Amazon Virtual Private Cloud: There's a separate pool of Elastic IP addresses to use with Amazon VPC. Your Amazon EC2 Elastic IP addresses won't work with instances in a VPC, and your VPC Elastic IP addresses won't work with instances in EC2. The following sections describe how to work with EC2 addresses, which have different characteristics than VPC Elastic IP addresses. For information about using VPC Elastic IP addresses, go to Elastic IP Addresses in the *Amazon Virtual Private Cloud User Guide*.

Elastic IP Address Concepts

By default, all Amazon EC2 instances are assigned two IP addresses at launch: a private (RFC 1918) address and a public address that is mapped to the private IP address through Network Address Translation (NAT).

If you use dynamic DNS to map an existing DNS name to a new instance's public IP address, it might take up to 24 hours for the IP address to propagate through the Internet. As a result, new instances might not receive traffic while terminated instances continue to receive requests.

To solve this problem, Amazon EC2 provides Elastic IP addresses. Elastic IP addresses are static IP addresses designed for dynamic cloud computing. Elastic IP addresses are associated with your account, not specific instances. Any Elastic IP addresses that you associate with your account remain associated with your account until you explicitly release them. Unlike traditional static IP addresses, however, Elastic IP addresses allow you to mask instance or Availability Zone failures by rapidly remapping your public IP addresses to any instance in your account.

You can associate one Elastic IP address with only one instance at a time in Amazon EC2. In Amazon VPC, you can associate Elastic IP addresses with a primary or secondary private IP address on an instance or network interface.

When you associate an Elastic IP address with an instance, its current public IP address is released to the Amazon EC2 public IP address pool. If you disassociate an Elastic IP address from the instance, the instance is automatically assigned a new public IP address within a few minutes. In addition, stopping the instance also disassociates the Elastic IP address from it. In Amazon VPC, when an instance is stopped, the Elastic IP address remains associated with the instance.

To ensure efficient use of Elastic IP addresses, we impose a small hourly charge when these IP addresses are not associated with a running instance, or when they are associated with a stopped instance or unattached network interface. AWS does not charge for one Elastic IP address while it is associated with a running instance. You are charged for additional Elastic IP addresses on an instance in Amazon VPC.

For an example of using Elastic IP addresses, see .

Elastic IP Addresses in Amazon VPC

VPC instances only have private IP addresses, so if you want an instance to communicate with the Internet, you must allocate an Elastic IP address for use with Amazon VPC and then assign that address to the instance. The address is a static, public IP address that you can assign to any instance or elastic network interface in your VPC. With an Elastic IP address, you can mask an instance failure by rapidly reassigning the address to another instance in your VPC.

Following are the basic things you need to know about Amazon VPC Elastic IP addresses:

- Any instance that needs to communicate with the Internet (that is, over the Internet gateway) must have an Elastic IP address associated with it.
- You first allocate an Elastic IP address for your VPCs, and then assign it to an instance in your VPC (it can be assigned to only one instance at a time).
- Elastic IP addresses you use in a VPC are different from ones you use outside a VPC (for a list of the differences, see the next section).
- You can move an Elastic IP address from one instance to another in the same VPC, or in any other VPCs that you are running, but not to instances outside the VPC.
- Any addresses you've allocated to your VPC remain with your VPC until you explicitly release them.
- To ensure efficient use of Elastic IP addresses, we impose a small hourly charge when these IP addresses are not associated with a running instance, or when they are associated with a stopped instance or an unattached network interface. You can associate an Elastic IP address to an elastic

network interface (ENI), however, if that ENI is not attached to a running instance, you will be charged for the Elastic IP address.

* You're limited to 5 VPC Elastic IP addresses; to help conserve them, you can use a NAT instance (see NAT Instances) in the *Amazon Virtual Private Cloud User Guide*. To request additional VPC Elastic IP Addresses, go to http://aws.amazon.com/contact-us/vpc-request/.

Differences Between EC2 Addresses and VPC Addresses

The following table lists the differences between EC2 Elastic IP addresses and those you can use in a VPC.

EC2	VPC
When you allocate an address, it's associated with your AWS account, but for use only outside a VPC.	When you allocate an address, it's associated with your AWS account, but for use only in a VPC.
If you try to associate an address that's already associated with another instance, the address is automatically associated with the new instance.	If you try to associate an address that's already associated with another instance, you can allow reassociation to reassociate the address.
If you stop an instance, its Elastic IP address is unmapped, and you must remap it when you restart the instance.	If you stop an instance, its Elastic IP address stays mapped.
Instances support only a single private IP address and a corresponding Elastic IP address.	Instances support multiple IP addresses and each one can have a corresponding Elastic IP address. For more information, see Multiple IP Addresses (p. 386).

API and CLI Overview

The following table summarizes the available Elastic IP address command line interface (CLI) tools commands and corresponding API actions. For more information about the CLI commands, go to the Amazon Elastic Compute Cloud Command Line Reference. For more information about the API actions, go to the Amazon Elastic Compute Cloud API Reference.

CLI Command and API Action	Description
`ec2-allocate-address` `AllocateAddress`	Acquires an Elastic IP address for use with your account.
`ec2-associate-address` `AssociateAddress`	Associates an Elastic IP address with an instance or a network interface or instances in a VPC.
`ec2-describe-addresses` `DescribeAddresses`	Lists Elastic IP addresses assigned to your account.
`ec2-disassociate-address` `DisassociateAddress`	Disassociates an Elastic IP address from the instance's specified IP address.
`ec2-release-address` `ReleaseAddress`	Releases an Elastic IP address associated with your account. After releasing an Elastic IP address, it is released to the IP address pool and might no longer be available to your account.

Allocating Elastic IP Addresses

This section describes how to assign an Amazon EC2 Elastic IP address to your account.

AWS Management Console

To allocate a new IP address for use with your account

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Click **Elastic IPs** in the **Navigation** pane.
   The console displays a list of Elastic IP addresses assigned to your account.
3. Click **Allocate New Address**.
   A confirmation dialog box appears.
4. Do one of the following

   - To use the Elastic IP address on an EC2 instance, ensure the value for EIP used in is set to EC2, and click **Yes, Allocate**.
   - To use the Elastic IP address on an instance running in an Amazon VPC, in **EIP**, select VPC for the value, and click **Yes, Allocate**.

   A new Elastic IP address appears in the list.

   Note

   The Elastic IP address is associated with your account and billed accordingly until you release it.

Command Line Tools

To allocate a new IP address for use with your account for an Amazon EC2 instance

- Enter the following command:

```
PROMPT> ec2-allocate-address
```

Amazon EC2 returns an Elastic IP address similar to the following:

```
ADDRESS 192.0.2.1    standard
```

The value `standard` means this address is for use only with standard (EC2) instances, and not Amazon VPC instances.

To allocate a new IP address for use with your account for an Amazon VPC instance

- Enter the following command:

```
PROMPT> ec2-allocate-address -d vpc
```

Amazon EC2 returns an Elastic IP address similar to the following:

```
ADDRESS   203.0.113.0   vpc       eipalloc-5723d13e
```

The value `vpc` means this address is for use only with instances running in Amazon VPC.

Note

The Elastic IP address is associated with your account and billed accordingly until you release the address with `ec2-release-address` command.

API

To allocate a new IP address for use with your account

- Construct the following Query request.

```
https://ec2.amazonaws.com/
?Action=AllocateAddress
&...auth parameters...
```

Following is an example response.

```
<AllocateAddressResponse xmlns="http://ec2.amazonaws.com/doc/2012-07-20/">
    <requestId>59dbff89-35bd-4eac-99ed-be587EXAMPLE</requestId>
    <publicIp>192.0.2.1</publicIp>
    <domain>standard</domain>
</AllocateAddressResponse>
```

The domain value `standard` means this address is for use only with standard (EC2) instances, and not Amazon VPC instances.

Note

The Elastic IP address is associated with your account and billed accordingly until you release the address with `ReleaseAddress`.

Describing Elastic IP Addresses

This section describes how to view the Elastic IP addresses allocated to your account.

AWS Management Console

To view Elastic IP addresses assigned to your account

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Click **Elastic IPs** in the **Navigation** pane.
   The console displays a list of Elastic IP addresses assigned to your account.
3. To reduce the size of the list, start typing part of the IP address or instance ID to which it is assigned in the search box.

Command Line Tools

To view Elastic IP addresses assigned to your account

1. To view all Elastic IP addresses assigned to your account:

   ```
   PROMPT> ec2-describe-addresses
   ```

   Amazon EC2 returns a list of Elastic IP addresses similar to the following:

   ```
   ADDRESS 203.0.113.0    standard
   ADDRESS 203.0.113.22   standard
   ADDRESS 203.0.113.44   vpc      eipalloc-5723d13e
   ```

   If the IP address is mapped, it is followed by the instance ID it's mapped to.

   ```
   ADDRESS  203.0.113.1  i-996fc0f2    standard
   ```

2. To verify a specific Elastic IP address:

   ```
   PROMPT> ec2-describe-addresses ip_address
   ```

   Amazon EC2 returns the specified Elastic IP address, similar to the following:

   ```
   ADDRESS 203.0.113.   standard
   ```

API

To view Elastic IP addresses assigned to your account

- Construct the following Query request.

  ```
  https://ec2.amazonaws.com/
  ?Action=DescribeAddresses
  &...auth parameters...
  ```

  Following is an example response.

  ```
  <DescribeAddressesResponse xmlns="http://ec2.amazonaws.com/doc/2012-07-20/">

     <requestId>59dbff89-35bd-4eac-99ed-be587EXAMPLE</requestId>
     <addressesSet>
       <item>
         <publicIp>203.0.113.2</publicIp>
         <domain>standard</domain>
         <instanceId/>
       </item>
     </addressesSet>
  </DescribeAddressesResponse>
  ```

Associating an Elastic IP Address with a Running Instance in Amazon EC2

After you allocate an Elastic IP address, you can map it to a running instance.

AWS Management Console

To associate an Elastic IP address with an instance

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Click **Instances** in the **Navigation** pane.
    The console displays a list of instances.
3. Write down the instance ID to associate with the Elastic IP address.
4. Click **Elastic IPs** in the **Navigation** pane.
    The console displays a list of Elastic IP addresses assigned to your account.
5. Select an address and click **Associate Address**.
    The **Associate Address** dialog box appears.
6. Select the instance from the **Instance** list box and click **Yes, Associate**.
    The Elastic IP address is associated with the instance.

Command Line Tools

To associate an Elastic IP address with an instance

1. Describe running instances:

    ```
    PROMPT> ec2-describe-instances
    ```

    Amazon EC2 returns output similar to the following:

    ```
    RESERVATION      r-ae33c2c7      111122223333    default
    INSTANCE i-afb130c4 ami-3c47a355 ec2-184-73-1-155.compute-1.amazonaws.com
    domU-12-31-39-09-1D-E5.compute-1.internal running gsg-keypair 0 m1.small
    2010-03-30T07:42:51+0000 us-east-1a aki-a71cf9ce ari-a51cf9cc monitoring-
    disabled    184.73.1.155    10.210.34.19   instance-store   paravirtual   xen
    RESERVATION   r-8a3374e2 111122223333   default
    INSTANCE i-85b435ee ami-b232d0db ec2-204-236-202-134.compute-1.amazonaws.com
     domU-12-31-39-00-86-35.compute-1.internal running gsg-keypair 0 m1.small
    2010-03-30T08:43:48+0000 us-east-1a aki-94c527fd ari-96c527ff monitoring-
    disabled   204.236.202.134 10.254.137.191  ebs    paravirtual   xen
    ```

    Write down the instance ID to associate with an Elastic IP address.
2. Describe Elastic IP addresses assigned to the account:

    ```
    PROMPT> ec2-describe-addresses
    ```

    Amazon EC2 returns a list of Elastic IP addresses similar to the following:

    ```
    ADDRESS 203.0.113.12    standard
    ADDRESS 198.51.100.1    standard
    ```

Write down the Elastic IP address to associate with an instance.

3. To associate the instance and Elastic IP address:

```
PROMPT> ec2-associate-address -i instance_id ip_address
```

Amazon EC2 returns output similar to the following:

```
ADDRESS 75.101.157.145    i-afb130c4
```

4. Associations take a few minutes to complete. To verify the association using
   `ec2-describe-addresses`:

```
PROMPT> ec2-describe-addresses
```

Amazon EC2 returns output similar to the following:

```
ADDRESS 75.101.157.145    i-afb130c4    standard
```

5. To verify the association using `ec2-describe-instances`:

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

Amazon EC2 returns output similar to the following:

```
RESERVATION r-9284a1fa 111122223333  default
INSTANCE i-afb130c4 ami-3c47a355 ec2-75-101-157-145.compute-1.amazonaws.com
 domU-12-31-39-09-25-62.compute-1.internal running 5fmorgin-ami    0
        m1.small    2010-03-17T13:17:41+0000        us-east-1a       aki-
a71cf9ce    ari-a51cf9cc      monitoring-disabled      75.101.157.145
10.210.42.144                    instance-store    paravirtual    xen
```

API

To associate an Elastic IP address with an instance

1. Describe running instances and write down the instance ID of the instance that you will associate
   with the Elastic IP address. For more information, see API (p. 385).
2. Describe Elastic IP addresses assigned to the account and write down the Elastic IP address that
   you will associate with the instance. For more information, see API (p. 406).
3. Associate the instance and Elastic IP address using the following Query request.

```
https://ec2.amazonaws.com/
?Action=AssociateAddress
&InstanceId=instance-id
&PublicIp=elastic-ip-address
&...auth parameters...
```

Following is an example response.

```
<AssociateAddressResponse xmlns="http://ec2.amazonaws.com/doc/2012-07-20/">

  <requestId>ce410ee1-068c-4026-a36e-86cefd453c28</requestId>
  <return>true</return>
</AssociateAddressResponse>
```

4.   Associations take a few minutes to complete. You can verify the association using `DescribeAddresses` or `DescribeInstances`.

Associating an Elastic IP Address with a Different Running Instance in Amazon EC2

Once an Elastic IP Address is allocated, you can map it to a different running instance.

Note

It is highly unlikely that an instance will be configured with its original public IP address that it used prior to being mapped.

AWS Management Console

To remap an IP address

1.   Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2.   Click **Instances** in the **Navigation** pane.
     The console displays a list of running instances.
3.   Write down the new instance ID to associate with the Elastic IP address.
4.   Click **Elastic IPs** in the **Navigation** pane.
     The console displays a list of Elastic IP addresses assigned to your account.
5.   Locate the IP address to remap and click **Disassociate**.
     A confirmation dialog box appears.
6.   Click **Yes, Disassociate**.
     The Elastic IP address is disassociated from the instance and you are returned to the list of Elastic IP addresses assigned to your account.
7.   Locate the IP address in the list and click **Associate**.
     The **Associate Address** dialog box appears.
8.   Select the new instance from the **Instance ID** list box and click **Associate**.
     The Elastic IP address is associated with the new instance.

Command Line Tools

To remap an IP address

1.   Describe running instances:

```
PROMPT> ec2-describe-instances
```

Amazon EC2 returns output similar to the following:

```
RESERVATION r-9284a1fa 111122223333  default
INSTANCE i-b2e019da ami-3c47a355 ec2-75-101-157-145.compute-1.amazonaws.com
 domU-12-31-39-09-25-62.compute-1.internal running gsg-keypair    0
        m1.small   2010-03-17T13:17:41+0000       us-east-1a      aki-
a71cf9ce   ari-a51cf9cc      monitoring-disabled    75.101.157.145
10.210.42.144                  instance-store   paravirtual   xen
RESERVATION      r-f25e6f9a      111122223333    default
INSTANCE       i-b2e019db ami-3c47a355    ec2-184-73-1-155.compute-
1.amazonaws.com  domU-12-31-39-09-1D-E5.compute-1.internal      running
gsg-keypair      0             m1.small   2010-03-30T07:42:51+0000
 us-east-1a     aki-a71cf9ce    ari-a51cf9cc     monitoring-disabled
  184.73.1.155    10.210.34.19                 instance-store   paravir
tual   xen
```

Write down the instance ID to associate with an Elastic IP address.

2. Associate the address with a new instance:

```
PROMPT> ec2-associate-address -i instance_id ip_address
```

Amazon EC2 returns output similar to the following:

```
ADDRESS 75.101.157.145   i-b2e019db   standard
```

3. Verify the changes:

```
PROMPT> ec2-describe-instances
```

Amazon EC2 returns output similar to the following:

```
RESERVATION r-9284a1fa 111122223333  default
INSTANCE i-b2e019da ami-3c47a355 ec2-184-73-1-123.compute-1.amazonaws.com
domU-12-31-39-09-25-62.compute-1.internal running gsg-keypair 0  m1.small
  2010-03-17T13:17:41+0000       us-east-1a aki-a71cf9ce ari-a51cf9cc mon
itoring-disabled  184.73.1.123 10.210.42.144  instance-store   paravirtual
   xen
RESERVATION  r-f25e6f9a 111122223333  default
INSTANCE i-b2e019db ami-3c47a355 ec2-75-101-157-145.compute-1.amazonaws.com
 domU-12-31-39-09-1D-E5.compute-1.internal running gsg-keypair 0  m1.small
 2010-03-30T07:42:51+0000 us-east-1a aki-a71cf9ce ari-a51cf9cc monitoring-
disabled  75.101.157.145 10.210.34.19  instance-store   paravirtual   xen
```

API

To remap an IP address

1. Describe running instances and write down the instance ID of the instance that you will associate with the Elastic IP address.

2. Describe Elastic IP addresses assigned to the account and write down the Elastic IP address to remap.

3. Associate the instance and Elastic IP address using the following Query request.

```
https://ec2.amazonaws.com/
?Action=AssociateAddress
&InstanceId=instance-id
&PublicIp=elastic-ip-address
&...auth parameters...
```

Following is an example response.

```
<AssociateAddressResponse xmlns="http://ec2.amazonaws.com/doc/2012-07-20/">

  <requestId>59dbff89-35bd-4eac-99ed-be587EXAMPLE</requestId>
  <return>true</return>
</AssociateAddressResponse>
```

Associations take a few minutes to complete.

4. Verify the association using `DescribeAddresses` or `DescribeInstances`.

Associating an Elastic IP Address in Amazon VPC

In Amazon VPC, you must associate an Elastic IP address with the private IP address of an instance if you want to allow traffic from the Internet. You can associate an Elastic IP address with both primary private IP addresses and secondary private IP addresses.

Important

Before you can successfully connect to an instance by using an Elastic IP address associated with a secondary private IP address, you must configure the operating system to use static IP addressing and to recognize the secondary private IP address. For more information, see Step 2: Configure the Operating System on Your Instance to Recognize the Secondary Private IP Address (p. 394).

AWS Management Console

To associate an Elastic IP address with to an instance or network interface in Amazon VPC

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Click **Instances** in the **Navigation** pane.
    The console displays a list of instances.
3. Write down the instance ID or network interface to associate with the Elastic IP address.
4. Click **Elastic IPs** in the **Navigation** pane.
    The console displays a list of Elastic IP addresses assigned to your account.
5. Select an address created for Amazon VPC, and then click **Associate Address**.
    The **Associate Address** dialog box appears.

6. In the **Associate Address** dialog box, in the Instance box, select the instance or network interface to which the private IP address is assigned.

7. In **Private IP Address**, select the private IP address.

8. Click **Allow Reassociation** to allow the Elastic IP address to be associated with the specified instance or network interface if it is currently associated with another instance or network interface..

   Note

   If the network interface or instance is already associated with an Elastic IP address, the operation will fail.

Command Line Tools

To associate an Elastic IP address with an instance or network interface

- Use the following command to associate an Elastic IP address with a private IP address on the specified instance in a VPC. The allow-reassociation option allows the Elastic IP address to be associated with the specified instance even if it is currently associated with another instance or network interface:

```
PROMPT> PROMPT>ec2-associate-address -a eipalloc-bf66dcd6 -i i-ba6a0dee  -
p 10
.0.0.85 --allow-reassociation
```

For more information, see ec2-associate-address in the *Amazon Elastic Compute Cloud Command Line Reference Guide*.

API

To use the API to associate an Elastic IP address with an instance or network interface in Amazon VPC, see AssociateAddress in the *Amazon Elastic Compute Cloud API Reference Guide*.

Example

The following diagram and table shows an example of using Elastic IP addresses.
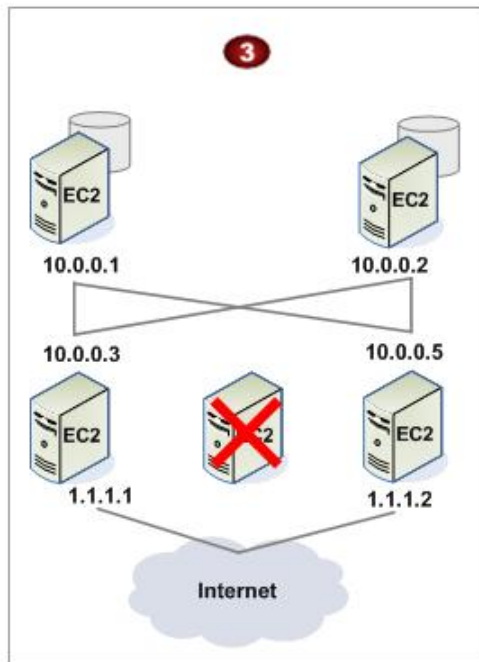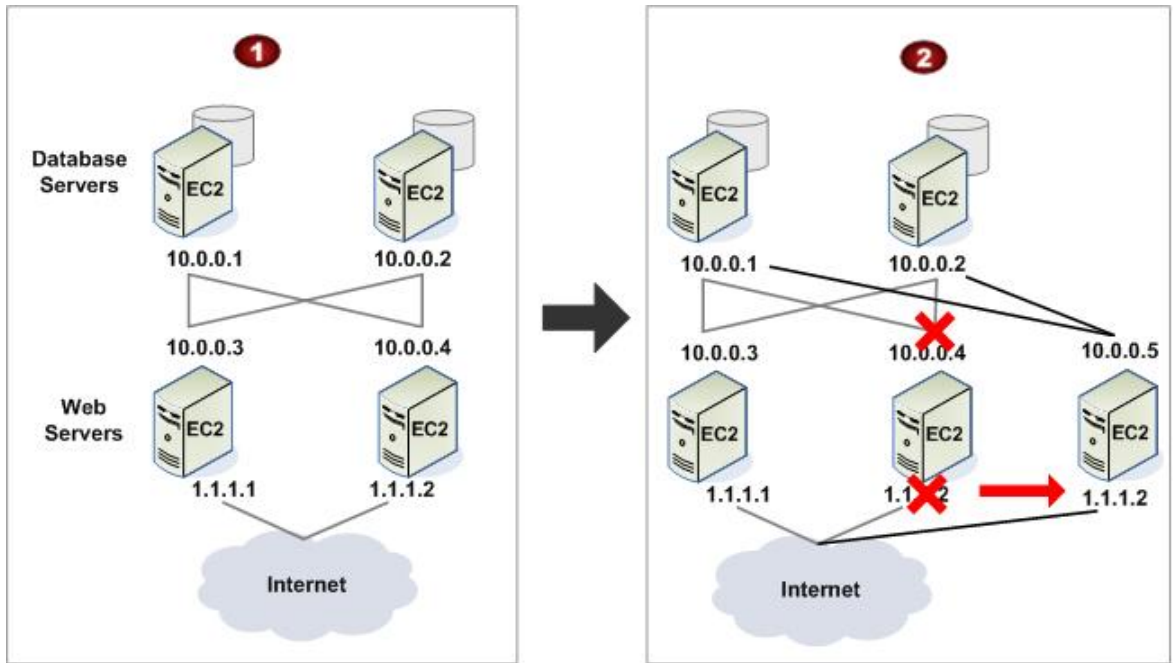
❶	Web servers are connected to the Internet through Elastic IP addresses (1.1.1.1 and 1.1.1.2) and to database servers through their private IP addresses.
❷	The administrator decides to replace a web server with a larger instance type. To do this, the administrator starts a new instance using a larger instance type, disassociates the 1.1.1.2 Elastic IP address from its running instance, and associates the address with the new instance.

3	The administrator terminates the old instance.

AWS Management Console

The following table describes how you set up the preceding tasks using the AWS Management Console.

Launch Process

1	Click **Instances** in the **Navigation** pane. The console displays a list of running instances.
2	Click **Launch Instance**, select **Classic Wizard** specify settings that meet the new requirements, and click **Launch**.
3	Click **Elastic IPs** in the **Navigation** pane. The console displays a list of Elastic IP addresses assigned to the account.
4	Select the desired IP address that is assigned to another instance, click **Disassociate**, and confirm the disassociation. You're returned to the list of Elastic IP addresses.
5	Locate the IP address in the list, click **Associate**, select the new instance, and confirm the association. The Elastic IP address is assigned to the new instance.
6	Click **Instances** in the **Navigation** pane, select the old instance, and clicks **Terminate**. Amazon EC2 begins shutting down the old instance.

Command Line Tools

The following example shows how to set up the preceding tasks using the command line tools (assuming the Elastic IP address is 203.0.113.01).

Note

The command to disassociate the address is optional because `ec2-associate-address` automatically disassociates the Elastic IP address if it is assigned to another instance.

```
PROMPT> ec2-run-instances ami-6ba54002 -n 1 --availability-zone us-east-1a
RESERVATION r-a034c7c9 924417782495 default
INSTANCE i-3ea74257 ami-6ba54002 pending 0 m1.large 2007-07-11T16:40:44+0000
us-east-1a

PROMPT> ec2-disassociate-address 203.0.113.01
ADDRESS 203.0.113.01

PROMPT> ec2-associate-address -i i-3ea74257 67.202.55.255
ADDRESS 203.0.113.01    i-43a4412a

PROMPT> ec2-terminate-instances i-4bc32334
INSTANCE i-4bc32334 running shutting-down
```

API

The following demonstrates how to set up these tasks using the API.

1. Run a new larger instance type using `RunInstances`.

2. Disassociate the Elastic IP address from the old instance using `DisassociateAddress`.

   Note

   This step is optional because `AssociateAddress` automatically disassociates the Elastic IP address if it is assigned to another instance.

3. Associate the Elastic IP address with the new instance using `AssociateAddress`.

4. Verify the association using `DescribeInstances`.

5. Terminate the old instance using `TerminateInstances`.

Elastic IP Address Limit

By default, all accounts are limited to 5 Elastic IP addresses because public (IPV4) Internet addresses are a scarce public resource. We strongly encourage you to use Elastic IPs primarily for load balancing use cases, and DNS hostnames for all other inter-node communication.

If you feel your architecture warrants an exception, please complete the Amazon EC2 Elastic IP Address Request Form. We will ask you to think through your use case and help us understand your need for additional addresses.

Using Reverse DNS for EMail Applications

If you intend to send email to third parties from Amazon EC2 instances, we suggest you provision one or more elastic IP addresses and provide them to us in the Request to Remove Email Sending Limitations form. AWS works with ISPs and Internet anti-spam organizations (such as Spamhaus) to reduce the chance that your email sent from these addresses will be flagged as spam.

In addition, assigning a static reverse DNS record to your Elastic IP address used to send email can help avoid having email flagged as spam by some anti-spam organizations. You can provide us with a reverse DNS record (for example: foo.yourcompany.com) to associate with your addresses through the aforementioned form. Note that a corresponding forward DNS record (A Record) pointing to your Elastic IP address must exist before we can create your reverse DNS record.

Security Groups

Topics

This section describes Amazon EC2 security groups and how to use them.

> **Important**
>
> For information about Amazon VPC security groups, go to Security Groups in the *Amazon Virtual Private Cloud User Guide*.

Security Group Concepts

A security group acts as a firewall that controls the traffic allowed into a group of instances. When you launch an Amazon EC2 instance, you can assign it to one or more security groups. For each security group, you add rules that govern the allowed inbound traffic to instances in the group. All other inbound traffic is discarded. You can modify rules for a security group at any time. The new rules are automatically enforced for all existing and future instances in the group.

> **Note**
>
> You can create up to 500 Amazon EC2 security groups in each region in an account, with up to 100 rules per security group. In Amazon VPC, you can have up to 50 security groups, with up to 50 rules per security group, in each VPC. The Amazon VPC security group limit does not count against the Amazon EC2 security group limit.

> **Caution**
>
> Because an Amazon EC2 instance can belong to multiple security groups, more than 100 rules can apply to an instance. Associating hundreds of rules with an instance might cause problems when you access the instance. We recommend you condense your rules as much as possible.

Default Security Group

Your AWS account automatically comes with a *default security group* for your Amazon EC2 instances. If you don't specify another security group at instance launch time, the instance is automatically launched into this default group. These are the initial settings for this group:

- Allow no inbound traffic
- Allow all outbound traffic
- Allow the instances in the group to talk to each other

You can change the default group's inbound rules. For example, you might change the rules to allow SSH or Remote Desktop connections from specific hosts for the purposes of instance management. For another example, see Modifying the Default Group (p. 427).

The default security group is automatically named *default*, and it has an AWS-assigned ID. You can't delete the default security group.

Creating Your Own Security Groups

If you don't want all your instances to use the default security group, you can create your own (up to 500 total). You can create groups that reflect the different roles your EC2 instances play in your system (e.g., web server, database server). For an example, see Creating a Three-Tier Web Service (p. 429).

When you create a new security group, you must provide a friendly name and a description of the group. AWS assigns each group a unique ID (e.g., sg-1a2b3c4d). These are the initial settings for a new group that you create:

- Allow no inbound traffic
- Allow all outbound traffic

After you've created the security group, you can change its rules to reflect the type of inbound traffic you want to allow into the group. You can change only the inbound rules; EC2 security groups don't have modifiable outbound rules. If you want the instances in the group to talk to each other, you must explicitly add rules to allow that.

Instance Group Membership

When you launch an instance, you can assign it to as many groups as you like. When deciding whether to allow traffic to a given instance, we evaluate all the rules from all the groups the instance is in.

If you don't specify any groups at launch time, the instance is automatically assigned to the *default* group and uses the rules you've set up for that group.

After an instance is running, you can't change which EC2 security groups it belongs to. However, you can change the rules of an existing group, and those changes propagate to the instances in the group.

Security Group Rules

An EC2 security group's rules control the inbound traffic allowed in to the instances in the group. All outbound traffic is automatically allowed. You can't change the outbound behavior.

Each EC2 security group rule enables a specific source to access the instances in the group using a certain protocol (TCP, UDP, or ICMP) and destination port or ports (if the protocol is TCP or UDP). For example, a rule could allow IP address 203.0.113.1 (the source) to access the instances in the group on TCP port 22 (the protocol and destination port). If you specify ICMP as the protocol for the rule, you must also specify an ICMP type and code.

The source can be an individual IP address (203.0.113.1), a range of addresses (e.g., 203.0.113.0/24), or an EC2 security group. The security group can be another group in your AWS account, a group in another AWS account, or the security group itself.

By specifying a security group as the source, you allow incoming traffic from all instances that belong to the source security group. The incoming traffic that you allow is based on the private IP addresses of the instances in the source security group. You might specify another security group in your account if you're creating a three-tier web service (see Creating a Three-Tier Web Service (p. 429)).

If you specify the security group itself as the source, each instance in the group will accept inbound traffic from fellow group members. For example, the default security group specifies itself as a source security group in its inbound security group rules. This is why members of the default security group allow inbound traffic from other members of the default security group.

You can't modify an existing rule in a group. However, you can add and remove rules to a group at any time. Your changes propagate to existing instances in the group after a short period.

Each EC2 security group can have up to 100 rules.

Understanding CIDR Notation

To specify a range of IP addresses using CIDR notation, you specify a base IP address and a suffix that indicates the number of significant bits in the IP address used to identify the network. The larger the suffix, the fewer number of hosts will be in the CIDR block. For example, say you specify a source as *10.10.1.32/27*. This specifies 10.10.1.32 as the base IP address and 27 as the suffix. Because 27 of the 32 possible bits in the IP4 address are allocated to the network address, there are only 5 (32 minus 27) bits available to specify host addresses. You can only represent 32 distinct numbers with 5 bits (2 to the 5th power = 32), which means 32 hosts on that network match the criteria and are in the CIDR block.

To better understand how this works, let's look at it in binary notation. The first 27 bits have to be an exact match in order for the range to match. It doesn't matter what happens after the 27 bits. For this example, any IP address that has 00001010.00001010.00000001.001 in the first 27 bits will match the address.

Here's how it works:

```
10.10.1.32 =
00001010.00001010.00000001.00100000
First 27 bits==
00001010.00001010.00000001.001
10.10.1.44 matches 10.10.1.32/27 =
00001010.00001010.00000001.00101100
But 10.10.1.90 doesn't match 10.10.1.32/27 =
00001010.00001010.00000001.01011010
```

For more information on CIDR, see http://en.wikipedia.org/wiki/Classless_Inter-Domain_Routing.

VPC Security Groups

If you're using Amazon Virtual Private Cloud, you must create *VPC security groups* specifically for your VPC instances. These groups work only inside the VPC where you've created them. Any EC2 security groups you have don't work inside your VPC. You can't create VPC security groups that reference EC2 groups, and you can't create EC2 security groups that reference VPC security groups..

You can have a VPC security group with the same name as an EC2 security group (it's possible because the groups have unique IDs). When working with VPC security groups, you must use the ID and not the name to identify the group.

VPC security groups have additional capabilities that EC2 security groups don't have. The following sections describe how to work with EC2 groups. To learn more about VPC security groups and how to work with them, go to Security Groups in the *Amazon Virtual Private Cloud User Guide*.

API and Command Overview

The following table summarizes the available commands and corresponding API actions for EC2 security groups. For more information about the commands, go to the Amazon Elastic Compute Cloud Command Line Reference. For more information about the API actions, go to the Amazon Elastic Compute Cloud API Reference.

Command and API Action	Description
`ec2-create-group` `CreateSecurityGroup`	Creates a new security group for use with your account.
`ec2-authorize` `AuthorizeSecurityGroupIngress`	Adds one or more rules to a security group.
`ec2-describe-group` `DescribeSecurityGroups`	Returns information about security groups associated with your account.
`ec2-revoke` `RevokeSecurityGroupIngress`	Removes one or more rules from a security group.
`ec2-delete-group` `DeleteSecurityGroup`	Deletes security groups associated with your account.

Note

VPC security groups have additional API actions and commands that apply to them. For more information, go to Security Groups in the *Amazon Virtual Private Cloud User Guide*.

Creating a Security Group

This section describes how to create a security group.

Note

You can create up to 500 Amazon EC2 security groups in each region in an account, with up to 100 rules per security group. In Amazon VPC, you can have up to 50 security groups, with up to 50 rules per security group, in each VPC. The Amazon VPC security group limit does not count against the Amazon EC2 security group limit.

Caution

Because an Amazon EC2 instance can belong to multiple security groups, more than 100 rules can apply to an instance. Associating hundreds of rules with an instance might cause problems when you access the instance. We recommend you condense your rules as much as possible.

AWS Management Console

To create a security group

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Click **Security Groups** in the **Navigation** pane.
   The console displays a list of current security groups.

3.   Click **Create Security Group**.
     The **Create Security Group** dialog box appears.

4.   Configure the following settings and click **Yes, Create**.

     - Name
     - Description
     - For **VPC**, select `No VPC`.

     Amazon EC2 creates the security group and adds it to your list of groups.

Command Line Tools

To create a security group

- Enter the following command:

```
PROMPT>  ec2-create-group groupname -d "group_description"
```

Amazon EC2 returns information similar to the following example.

```
GROUP    sg-43de5aab    webservers    My Web Server Group
```

API

To create a security group

- Construct a Query request similar to the following example.

```
https://ec2.amazonaws.com/
?Action=CreateSecurityGroup
&GroupName=security-group-name
&GroupDescription=security-group-description
&...auth parameters...
```

Following is an example response.

```
<CreateSecurityGroupResponse xmlns="http://ec2.amazonaws.com/doc/2012-07-
20/">
  <requestId>53effa54-17f1-47ae-ba5c-118b1d2617c2</requestId>
  <return>true</return>
  <groupdId>sg-43de5aab</groupId>
</CreateSecurityGroupResponse>
```

Describing Security Groups

This section describes how to view your security groups.

AWS Management Console

To view security groups

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Click **Security Groups** in the **Navigation** pane.
   The console displays a list of security groups that belong to the account.
3. To view more information about a security group, including its rules, select it.
   The group's information is displayed in the lower pane.

Command Line Tools

To view security groups

- Enter the following command:

```
PROMPT>  ec2-describe-group [group ...]
```

Amazon EC2 returns output similar to the following example.

```
GROUP    sg-455b6c31      111122223333     WebServers    web
PERMISSION        111122223333     WebServers ALLOWS   tcp      80      80
 FROM    CIDR    0.0.0.0/0        ingress
```

Tip

You can filter this list to return only certain security groups of interest to you. For more information about how to filter the results, go to ec2-describe-group in the *Amazon Elastic Compute Cloud Command Line Reference.*

API

To view security groups

- Construct the following Query request.

```
https://ec2.amazonaws.com/
?Action=DescribeSecurityGroups
&GroupName.1=security-group-name
&...auth parameters...
```

Following is an example response.

```
<DescribeSecurityGroupsResponse xmlns="http://ec2.amazonaws.com/doc/2012-
07-20/">
  <requestId>59dbff89-35bd-4eac-99ed-be587EXAMPLE>/requestId>
  <securityGroupInfo>
    <item>
      <ownerId>999988887777</ownerId>
      <groupId>sg-455b6c31</groupId>
```

```
        <groupName>WebServers</groupName>
        <groupDescription>Web</groupDescription>
        <vpcId/>
        <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>
        <ipPermissionsEgress/>
        <tagSet/>
    </item>
  </securityGroupInfo>
</DescribeSecurityGroupsResponse>
```

Tip

You can filter this list to return only certain security groups of interest to you. For more information about how to filter the results, go to DescribeSecurityGroups in the *Amazon Elastic Compute Cloud API Reference*.

Adding a Security Group Rule

This section describes how to add a rule to a security group.

When you add a rule to a security group, the new rule is automatically applied to any instances in the group.

Note

You can create up to 500 Amazon EC2 security groups in each region in an account, with up to 100 rules per security group. In Amazon VPC, you can have up to 50 security groups, with up to 50 rules per security group, in each VPC. The Amazon VPC security group limit does not count against the Amazon EC2 security group limit.

Caution

Because an Amazon EC2 instance can belong to multiple security groups, more than 100 rules can apply to an instance. Associating hundreds of rules with an instance might cause problems when you access the instance. We recommend you condense your rules as much as possible.

Note

After an EC2 instance is running, you can't change which EC2 security groups it belongs to.
Exception: If you're using Amazon VPC, you can change which VPC security groups a VPC instance is in after launch. For more information, go to Security Groups in the *Amazon Virtual Private Cloud User Guide*.

AWS Management Console

To add a rule to a security group

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Click **Security Groups** in the **Navigation** pane.

    The console displays a list of security groups that belong to the account.
3. Select an EC2 security group.

    Its rules appear on the **Inbound** tab in the lower pane.

4. To add a rule:

    a. From the **Create a new rule:** drop-down list, select the option you want.

    b. Specify a port or port range (if you've chosen a custom protocol rule).

    c. In the **Source** field, specify one of the following:

    - Name or ID of a security group (to allow access from that group). If the group isn't in your AWS account, prefix the group name with the AWS account ID and a forward slash (e.g., 111122223333/OtherSecurityGroup).

    - IP address range in CIDR notation (to allow access from that IP address range). For example, enter 0.0.0.0/0 to allow all IP addresses to access the specified port range. Enter an IP address or range of addresses to limit access to one computer or a network, for example 203.0.113.5/32.

5. Click **Add Rule**.

    An asterisk appears on the **Inbound** tab.
6. Click **Apply Rule Changes**.

    The new rule is created and applied to all instances that belong to the security group.

Command Line Tools

To add a rule to a security group

- Enter the following command:

```
PROMPT> ec2-authorize group [-P protocol] (-p port_range | -t icmp_type_code)
 [-u source_group_user ...] [-o source_group ...] [-s source_subnet ...]
```

For example, to modify the default security group to allow port 80 access from all IP addresses:

```
PROMPT>  ec2-authorize default -p 80
```

Amazon EC2 returns output similar to the following example.

```
GROUP    sg-2eac845a
PERMISSION                      ALLOWS   tcp     80      80      FROM    CIDR
    0.0.0.0/0       ingress
```

For example, to modify the default security group to allow SSH access from your own system's IP address:

```
PROMPT>  ec2-authorize -p 22 -s <your_ip_address>/32
```

Amazon EC2 returns output similar to the following example.

```
GROUP    sg-2eac845a
PERMISSION                      ALLOWS   tcp     22      22      FROM    CIDR
    <your_ip_address>/32 ingress
```

API

To add a rule to a security group

- Construct a Query request similar to the following.

```
https://ec2.amazonaws.com/
?Action=AuthorizeSecurityGroupIngress
&IpPermissions.1.IpProtocol=tcp
&IpPermissions.1.FromPort=80
&IpPermissions.1.ToPort=80
&IpPermissions.1.IpRanges.1.CidrIp=0.0.0.0/0
&...auth parameters...
```

Following is an example response.

```
<AuthorizeSecurityGroupIngressResponse xmlns="http://ec2.amazon
aws.com/doc/2012-07-20/">
  <requestId>59dbff89-35bd-4eac-99ed-be587EXAMPLE</requestId>
```

```
  <return>true</return>
</AuthorizeSecurityGroupIngressResponse>
```

Deleting a Security Group Rule

This section describes how to delete a security group rule.

When you delete a rule from a security group, the change is automatically applied to any instances in the group.

AWS Management Console

To delete a security group rule

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Click **Security Groups** in the **Navigation** pane.
   The console displays a list of security groups that belong to the account.
3. Select a security group.
   Its rules appear on the **Inbound** tab in the lower pane.
4. To delete a rule, click **Delete** next to the rule.
   An asterisk appears on the **Inbound** tab.
5. Click **Apply Rule Changes**.
   Amazon EC2 deletes the security group rule.

Command Line Tools

The syntax of the `ec2-revoke` command is essentially identical to the syntax of the `ec2-authorize` command. To delete a rule, you specify the parts of the rule (e.g., protocol, port, CIDR range, etc.).

To delete a security group rule

- Enter the following command:

```
PROMPT>  ec2-revoke
group
[-P protocol]
(-p port_range | -t icmp_type_code)
[-u source_group_user ...]
[-o source_group ...]
[-s source_subnet ...]
```

For example, to modify the default security group to remove port 80 access from all IP addresses:

```
PROMPT>  ec2-revoke default -p 80
```

Amazon EC2 returns output similar to the following example.

```
GROUP    sg-2eac845a
PERMISSION                            ALLOWS   tcp      80       80       FROM    CIDR
    0.0.0.0/0          ingress
```

You can confirm the rule has been deleted by describing the security group with `ec2-describe-group`.

API

To delete a security group rule

- Construct a Query request similar to the following.

```
https://ec2.amazonaws.com/
?Action=RevokeSecurityGroupIngress
&IpPermissions.1.IpProtocol=tcp
&IpPermissions.1.FromPort=80
&IpPermissions.1.ToPort=80
&IpPermissions.1.IpRanges.1CidrIp=0.0.0.0/0
&...auth parameters...
```

Following is an example response.

```
<RevokeSecurityGroupIngressResponse xmlns="http://ec2.amazonaws.com/doc/2012-
07-20/">
   <requestId>59dbff89-35bd-4eac-99ed-be587EXAMPLE</requestId>
   <return>true</return>
</RevokeSecurityGroupIngressResponse>
```

Deleting a Security Group

This section describes how to delete a security group.

The group must not have any instances in it. You can't delete the default security group.

AWS Management Console

To delete a security group

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Click **Security Groups** in the **Navigation** pane.
   The console displays a list of security groups that belong to the account.
3. Select a security group and click **Delete**.
   A confirmation dialog box appears.
4. Click **Yes, Delete**.
   Amazon EC2 deletes the security group.

Command Line Tools

To delete a security group

- Enter the following command:

```
PROMPT>  ec2-delete-group group
```

Amazon EC2 returns output similar to the following:

```
RETURN true
```

API

To delete a security group

- Construct the following Query request.

```
https://ec2.amazonaws.com/
?Action=DeleteSecurityGroup
&GroupName=security-group-name
&...auth parameters...
```

Following is an example response.

```
<DeleteSecurityGroupResponse xmlns="http://ec2.amazonaws.com/doc/2012-07-
20/">
  <requestId>59dbff89-35bd-4eac-99ed-be587EXAMPLE</requestId>
  <return>true</return>
</DeleteSecurityGroupResponse>
```

Examples

This section provides examples of configuring EC2 security groups using the command line tools.

Note

In addition to these examples, 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.

Modifying the Default Group

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

Albert Modifies the Default Group

1	Albert launches a copy of his favorite public AMI.
| | ```
PROMPT> ec2-run-instances ami-eca54085
RESERVATION r-a034c7c9 111122223333 default
INSTANCE i-cfd732a6 ami-eca54085 pending 0 m1.small 2010-03-
19T13:59:03+0000 us-east-1a aki-94c527fd ari-96c527ff monitoring-
disabled  ebs
``` |
| 2 | Albert, who is a cautious type, checks the access rules of the default group. |
| | ```
PROMPT> ec2-describe-group default
GROUP   sg-2eac845a     111122223333    default default group
PERMISSION      111122223333    default ALLOWS  icmp    -1      -1
    FROM    USER    111122223333    NAME default    ID sg-2eac845a
ingress
PERMISSION      111122223333    default ALLOWS  tcp     0       65535
    FROM    USER    111122223333    NAME default    ID sg-2eac845a
ingress
PERMISSION      111122223333    default ALLOWS  udp     0       65535
    FROM    USER    111122223333    NAME default    ID sg-2eac845a
ingress
``` |
| | Albert notices that it only accepts ingress network connections from other members of the default group for all available protocols (TCP, UDP, ICMP) and ports. |
| 3 | Albert, being cautious, uses the Linux/UNIX nmap command to port scan his instance. |
| | ```
$ nmap -P0 -p1-100 ec2-203-0-113-5.compute-1.amazonaws.com
Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2010-03-30
15:06 SAST
All 100 scanned ports on ec2-203-0-113-5.compute-1.amazonaws.com
(203.0.113.5) 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. |
| | ```
PROMPT> ec2-authorize default -P tcp -p 22 -s 203.0.113.7/32
GROUP   sg-2eac845a
PERMISSION    ALLOWS  tcp  22  22  FROM  CIDR  203.0.113.7/32  ingress
``` |

5	Albert repeats the Linux/UNIX `nmap` port scan.
| | ```
$ nmap -P0 -p1-100 ec2-203-0-113-5.compute-1.amazonaws.com
Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2010-03-30
15:07 SAST
Interesting ports on ec2-203-0-113-5.compute-1.amazonaws.com
(203.0.113.5):
(The 99 ports scanned but not shown 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. |

Creating a Three-Tier Web Service

Mary wants to deploy her public, failure resilient, three-tier web service (web, application, and database servers) 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.

Launch Process

1	First, Mary creates a group for her Apache web server instances and allows HTTP access to the world.
| | ```
PROMPT> ec2-create-group apache -d "Mary's Apache group"
GROUP   sg-ed5b6c99   apache   Mary's Apache group

PROMPT> ec2-describe-group apache
GROUP   sg-ed5b6c99   111122223333    apache  Mary's Apache group

PROMPT> ec2-authorize apache -P tcp -p 80 -s 0.0.0.0/0
GROUP   sg-ed5b6c99
PERMISSION  111122223333 apache  ALLOWS  tcp   80   80    FROM
CIDR    0.0.0.0/0   ingress

PROMPT> ec2-describe-group apache
GROUP   sg-ed5b6c99  111122223333    apache  Mary's Apache group
PERMISSION   111122223333   apache  ALLOWS  tcp   80   80   FROM
CIDR    0.0.0.0/0   ingress
``` |

2	Mary launches seven instances of her web server AMI as members of the `apache` group.
|   | ```
PROMPT> ec2-run-instances ami-fba54092 -n 7 -g apache
RESERVATION r-0592776c 111122223333 apache
INSTANCE i-cfd732a6 ami-fba54092 pending 0 m1.small 2010-03-
19T13:59:03+0000 us-east-1a aki-94c527fd ari-96c527ff monitoring-
disabled ebs   paravirtual   xen   sg-ed5b6c99
INSTANCE i-cfd732a7 ami-fba54092 pending 1 m1.small 2010-03-
19T13:59:03+0000 us-east-1a aki-94c527fd ari-96c527ff monitoring-
disabled ebs   paravirtual   xen   sg-ed5b6c99
INSTANCE i-cfd732a8 ami-fba54092 pending 2 m1.small 2010-03-
19T13:59:03+0000 us-east-1a aki-94c527fd ari-96c527ff monitoring-
disabled ebs   paravirtual   xen   sg-ed5b6c99
INSTANCE i-cfd732a9 ami-fba54092 pending 3 m1.small 2010-03-
19T13:59:03+0000 us-east-1a aki-94c527fd ari-96c527ff monitoring-
disabled ebs   paravirtual   xen   sg-ed5b6c99
INSTANCE i-cfd732aa ami-fba54092 pending 4 m1.small 2010-03-
19T13:59:03+0000 us-east-1a aki-94c527fd ari-96c527ff monitoring-
disabled ebs   paravirtual   xen   sg-ed5b6c99
INSTANCE i-cfd732ab ami-fba54092 pending 5 m1.small 2010-03-
19T13:59:03+0000 us-east-1a aki-94c527fd ari-96c527ff monitoring-
disabled ebs   paravirtual   xen   sg-ed5b6c99
INSTANCE i-cfd732ac ami-fba54092 pending 6 m1.small 2010-03-
19T13:59:03+0000 us-east-1a aki-94c527fd ari-96c527ff monitoring-
disabled ebs   paravirtual   xen   sg-ed5b6c99


PROMPT> ec2-describe-instances i-cfd732a6
RESERVATION     r-0592776c      111122223333  apache
INSTANCE        i-cfd732a6      ami-fba54092       ec2-203-0-113-
12.compute-1.amazonaws.com       running 0
m1.small  2010-03-30T08:43:48+0000       us-east-1a    aki-94c527fd
    ari-96c527ff    monitoring-disabled    203.0.113.12
10.254.137.191                  ebs  paravirtual  xen  sg-ed5b6c99
BLOCKDEVICE    /dev/sda1      vol-cf13b3a6    2010-03-
30T08:01:44.000Z
``` |
| 3 | Being as cautious as Albert, Mary uses the Linux/UNIX `nmap` command to confirm the permissions she just configured. |
|   | ```
$ nmap -P0 -p1-100 ec2-203-0-113-12.compute-1.amazonaws.com
Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2010-03-30
15:10 SAST
Interesting ports on ec2-203-0-113-12.compute-1.amazonaws.com
(203.0.113.12):
(The 99 ports scanned but not shown 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 ec2-203-0-113-12.compute-1.amazonaws.com  80
Trying 203.0.113.12...
Connected to ec2-203-0-113-12.compute-1.amazonaws.com  (203.0.113.12).
Escape character is '^]'.
``` |
| | Mary can reach her web server. |
| 5 | Mary creates a separate group for her application server. |
| | ```
PROMPT> ec2-create-group appserver -d "Mary's app server"
GROUP   sg-e95b6c9d     appserver       Mary's app server
``` |
| 6 | Mary starts twenty instances as members of `appserver` group. |
| | ```
PROMPT> ec2-run ami-e3a5408a -n 20 -g appserver
``` |
| 7 | Mary grants network access between her web server group and the application server group. |
| | ```
PROMPT> ec2-authorize appserver -o apache -u 111122223333
GROUP                     appserver
PERMISSION                appserver       ALLOWS  tcp    0      65535
   FROM    USER   111122223333   GRPNAME apache            ingress
PERMISSION                appserver       ALLOWS  udp    0      65535
   FROM    USER   111122223333   GRPNAME apache            ingress
PERMISSION                appserver       ALLOWS  icmp   -1     -1
    FROM    USER   111122223333   GRPNAME apache           ingress
``` |
| 8 | Mary verifies access to her app server is restricted by port scanning one of the application servers using the Linux and UNIX `nmap` command. |
| | ```
$ nmap -P0 -p1-100 ec2-203-0-113-9.compute-1.amazonaws.com
Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2010-03-30
15:11 SAST
All 100 scanned ports on ec2-203-0-113-9.compute-1.amazonaws.com
(203.0.113.9) 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:
| | ```
PROMPT> ec2-authorize apache -P tcp -p 22 -s 203.0.113.19/32
``` |
| | B. She logs in to one of her web servers and connects to an application server on TCP port 8080. |
| | ```
$ telnet ec2-203-0-113-9.compute-1.amazonaws.com  8080
Trying 203.0.113.9...
Connected to ec2-203-0-113-9.compute-1.amazonaws.com (203.0.113.9).
Escape character is '^]'
``` |
| | C. Satisfied with the setup, she revokes SSH access to the web server group. |
| | ```
PROMPT> ec2-revoke apache -P tcp -p 22 -s 203.0.113.19/32
``` |
| 10 | Mary repeats these steps to create the database server group and to grant access between the application server and database server groups. |

Storage

Topics

Amazon EC2 provides you with flexible, cost effective, and easy-to-use data storage options for your instances. Each option has a unique combination of performance and durability. These storage options can be used independently or in combination to suit your requirements.

After reading this section, you should have a good understanding about how you can use the data storage options supported by Amazon Elastic Compute Cloud to meet your specific requirements. These storage options include the following:

- Amazon Elastic Block Store (Amazon EBS)
- Amazon EC2 instance store
- Amazon Simple Storage Service (Amazon S3)

Amazon EBS

Amazon EBS is a durable, block-level storage volume that you can attach to a single, running Amazon EC2 instance. You can use Amazon EBS as a primary storage device for data that requires frequent and granular updates. For example, Amazon EBS is the recommended storage option when you run a database on an instance.

Amazon EBS volumes behave like raw, unformatted, external block devices that you can attach to your instances. They persist independently from the running life of an Amazon EC2 instance. After an Amazon EBS volume is attached to an instance, you can use it like any other physical hard drive. For more information, see Amazon Elastic Block Store (Amazon EBS) (p. 434).

Amazon EC2 Instance Store

Each Amazon EC2 instance, unless it's a micro instance, can access storage from disks that are physically attached to the host computer. This disk storage is referred to as instance store. Instance store provides temporary block-level storage for Amazon EC2 instances. The data on an instance store volume persists only during the life of the associated Amazon EC2 instance. For more information, see Amazon EC2 Instance Store (p. 479).

Amazon S3

Amazon S3 is a repository for Internet data. Amazon S3 provides access to reliable and inexpensive data storage infrastructure. It is designed to make web-scale computing easier by enabling you to store and retrieve any amount of data, at any time, from within Amazon EC2 or anywhere on the web. For example, you can use Amazon S3 to store backup copies of your data and applications. For more information, see Amazon Simple Storage Service (Amazon S3) (p. 487).

Adding Storage

Every time you launch an instance from an AMI, a root storage device is created for that instance. The root storage device contains all the information necessary to boot the instance. For more information, see Root Device Volume (p. 117). You can specify storage volumes in addition to the root device volume when you create an AMI or launch an instance using *block device mapping*. For more information, see Block Device Mapping (p. 488).

You can also attach EBS volumes to a running instance. For more information, see Attaching a Volume to an Instance (p. 450).

Amazon Elastic Block Store (Amazon EBS)

Topics

Amazon Elastic Block Store (Amazon EBS) provides block level storage volumes for use with Amazon EC2 instances. Amazon EBS volumes are highly available and reliable storage volumes that can be attached to any running instance in the same Availability Zone. The Amazon EBS volumes attached to an Amazon EBS instance are exposed as storage volumes that persist independently from the life of the instance.

Amazon EBS is recommended when data changes frequently and requires long-term persistence. Amazon EBS is particularly well-suited for use as the primary storage for a file system, database, or for any applications that require fine granular updates and access to raw, unformatted block-level storage. Amazon EBS is particularly helpful for database-style applications that frequently encounter many random reads and writes across the dataset.

You can attach multiple volumes to the same instance within the limits specified by your AWS account. As per your AWS account, you can have up to 5000 EBS volumes or 20 TiB in total volume storage, whichever you reach first. Each EBS volume can be from 1GiB to 1 TiB. If you find that the default volume limits don't meet your specific requirements, go to the Amazon EBS Volume Limit Request Form to request different volume limits.

Key Features of EBS Volumes

All Amazon EBS volumes offer the following features:

- Data availability from replication across an Availability Zone
- Data persistence independent of the life of the instance
- The ability to create snapshots and incremental backups

Data Availability

When you create an Amazon EBS volume in an Availability Zone, it is automatically replicated within that zone to prevent data loss due to failure of any single hardware component. After you create a volume, you can attach it to any Amazon EC2 instance in the same Availability Zone. After you attach a volume, it appears as a native block device similar to a hard drive or other physical device. At that point, the instance can interact with the volume just as it would with a local drive; the instance can format the EBS volume with a file system such as ext3 (Linux) or NTFS (Windows) and install applications. You use the file system to access the stored files.

An EBS volume can be attached to only one instance at a time within the same Availability Zone. However, multiple volumes can be attached to a single instance. If you attach multiple volumes to a device that you have named, you can stripe data across the volumes for increased I/O and throughput performance.

You can get monitoring data for your Amazon EBS volumes at no additional charge (this includes data for the root device volumes for Amazon EBS-backed instances). For more information, see Monitoring Amazon EBS Volumes (p. 289).

Data Persistence

An Amazon EBS volume is off-instance storage that can persist independently from the life of an instance. You continue to pay for the volume usage as long as the data persists.

By default, EBS volumes that are attached to a running instance automatically detach from the instance with their data intact when that instance is terminated. The volume can then be reattached to a new instance, enabling quick recovery. If you are using an EBS-backed instance, you can stop and restart that instance without affecting the data stored in the attached volume. The volume remains attached throughout the stop-start cycle. This enables you to process and store the data set indefinitely, only using the processing and storage resources when required. The data set persists on the volume until the volume is deleted explicitly. After a volume is deleted, it can't be attached to any instance.

By default, EBS volumes that are created and attached to an instance at launch are deleted when that instance is terminated. You can modify this behavior by changing the value of the flag `DeleteOnTermination` to `false` when you launch the instance. This modified value causes the volume to persist even after the instance is terminated, and enables you to attach the volume to another instance.

Snapshots

Amazon EBS provides the ability to create snapshots (backups) of any Amazon EC2 volume. The volume need not be attached to a running instance in order to take a snapshot. Creating a snapshot of an EBS volume writes a copy of the data in the volume to Amazon S3, where it is stored redundantly in multiple Availability Zones. As you continue to write data to a volume, you can periodically create a snapshot of the volume to use as a baseline for new volumes. These snapshots can be used to create multiple new

Amazon EBS volumes, expand the size of a volume, or move volumes across Availability Zones. When you create a new volume using a snapshot, it's an exact replica of the original volume. By optionally specifying a different volume size or a different Availability Zone, you can use this functionality to increase the size of an existing volume or to create duplicate volumes in new Availability Zones. The snapshots can be shared with specific AWS accounts or made public.

Amazon EBS snapshots are incremental backups, meaning that only the blocks on the device that have changed since your last snapshot will be saved. If you have a device with 100 GiB of data, but only 5 GiB of data have changed since your last snapshot, only the 5 GiB of modified data is written to Amazon S3. Even though snapshots are saved incrementally, the snapshot deletion process is designed so that you need to retain only the most recent snapshot in order to restore the volume.

To help categorize and manage your volumes and snapshots, you can tag them with metadata of your choice. For more information, see Tagging Your Resources (p. 508).

EBS Volume Types

There are two types of EBS volumes:

- Standard
- Provisioned IOPS (input/output operations per second)

Standard EBS Volumes

Standard volumes offer cost effective storage that is ideal for applications with light or bursty I/O requirements.

These volumes deliver approximately 100 IOPS on average, with burst capability of up to hundreds of IOPS.

Standard volumes are well suited for use as boot volumes, as their burst capability improves instance start-up times.

Provisioned IOPS Volumes

To maximize the performance of your I/O-intensive applications, you can use Provisioned IOPS volumes. Provisioned IOPS volumes are designed to meet the needs of I/O-intensive workloads, particularly database workloads, that are sensitive to storage performance and consistency in random access I/O throughput. You specify an IOPS rate when you create the volume, and Amazon EBS provisions that rate for the lifetime of the volume. Amazon EBS currently supports up to 1,000 IOPS per volume. You can stripe multiple volumes together to deliver thousands of IOPS per instance to your application.

A Provisioned IOPS volume must be at least 10 GB in size. The ratio of IOPS provisioned to the volume size requested can be a maximum of 10. For example, a volume with 1000 IOPS must be at least 100 GB.

Provisioned IOPS volumes are designed to offer consistent high performance. They deliver within 10 percent of the provisioned IOPS performance 99.9 percent of the time over a given year, if the following guidelines are met:

- The volume is attached to an EBS-Optimized instance. For more information, see EBS-Optimized Instances (p. 88).
- The average queue length is at least 1 per 200 IOPS provisioned.
- The read and write operations have a block size of 16 KB or less. If the I/O increases above 16 KB, the IOPS delivered drop in proportion to the increase in the size of the I/O. For example, a 1000 IOPS volume can deliver 1000 16 KB writes per second, 500 32 KB writes per second, or 250 64 KB writes per second.

As with standard EBS volumes, there is up to a 50 percent reduction in IOPS when you first access the data. Performance is restored after the data is accessed once. For maximum performance consistency with new volumes, we recommend that you read or write to all the blocks on your volume before you use it for a workload.

Note that taking a snapshot can reduce the rate of IOPS you get from your volume while the snapshot is pending. To minimize the impact of snapshots on performance of a master node, create snapshots from a read replica of your data. Ideally, create these snapshots during off-peak usage.

For more information about volume performance, see Increasing EBS Performance (p. 439).

Common Tasks

This section describes some of the common tasks performed when using Amazon EBS.

Expanding the Storage Space of a Volume

1	Create a snapshot of the volume attached to your running instance.
2	Create a new volume from the snapshot by specifying a larger size than the original volume.
3	Attach the new volume to your instance.
4	Detach and delete the old volume.

Adding Multiple Copies of Your Volume to Your EC2 Instances

1	Create a snapshot of the volume attached to your running instance.
2	Create the new volumes you need using the snapshot.
3	Attach the newly created volumes to your EC2 instances.

Reducing Use of Storage Resources When Traffic Decreases

1	Identify the volume or volumes that are not currently needed.
2	[optional] Create snapshots of the identified volumes. Any AWS Marketplace product codes from the volume are propagated to the snapshot.
3	Detach the volume(s) from the instance. **Note** The volume data persists in the detached volume. You can attach a detached volume to any running instance.
4	To remove the data from the volume, delete the volume. **Note** The data in this volume is deleted. You can't attach this volume to any instance.

Data Persistence after Instance Termination

By default, any volumes that you attached when the instance is launched are automatically deleted when the instance terminates. However, any volumes that you attached to a running instance persist even after the instance terminates. You can change the default behavior using the `DeleteOnTermination` flag. For an example of how to change this flag when launching an instance, see Using Amazon EC2 Root Device Storage (p. 122).

The following diagram and task list describe the different states of a volume from the time it is attached to an instance until the time the instance terminates.

Persisting Data after Instance Termination

1	Launch an instance from an EBS-backed AMI and set the value of `DeleteOnTermination` flag to `false.` The volume is in its initial state.
2	Create snapshots of the volume at regular intervals.
3	Make changes to the data in the volume by doing some processing. The volume is now in a modified state.
4	Terminate the instance. The volume is detached from the instance.

The volume, after instance termination, persists in its modified state. This is a new and different volume; it's no longer in its initial state or associated with its original AMI.

When you launch an instance of a particular AMI and then terminate the instance, the detached volume retains its original volume ID. If you launch another instance of the same AMI, the volume associated with the newly launched instance has a different volume ID.

Using the Modified Volume (Volume B) after Instance Termination

1	Create a new EBS-backed AMI from the snapshot of the detached volume.
2	Launch an instance using the newly created AMI. The modified (Volume B) volume is attached to the instance.

Or

Attach the modified volume to another instance.

Using the Initial Volume (Volume A) after Instance Terminates

Launch an instance from EBS-backed AMI and set the value of `DeleteOnTermination` flag to `false`.

> **Note**
>
> The instance launches with the EBS volume (Volume A) in its initial state with a new volume ID.

Data Availability

A single Amazon EBS volume is constrained to the single Availability Zone where it is created and becomes unavailable if the Availability Zone itself is unavailable. However, a snapshot of an Amazon EBS volume is available across all of the Availability Zones within a Region. You can use these snapshots to create volumes and make them available in more than one Availability Zone.

Making Your Data Available in Multiple Availability Zones

1	Create a snapshot of the volume that you want to make available in another Availability Zone.
2	Create a new volume using the snapshot created in the previous step, and specify a different Availability Zone.
3	Launch a new instance in that Availability Zone and attach the new volume.

Increasing EBS Performance

After you have learned the basics of working with EBS volumes, it's a good idea to look at the I/O performance you require and your options for increasing EBS performance to meet those requirements.

Topics

Maximize EBS Volume Performance

To ensure the best performance from your EBS volumes:

- Use striping for multiple volumes
- Maintain the recommended average queue length
- Read or write to all blocks on the volume before it is first used

Striping Options for EBS Volumes

You can use RAID 0 or RAID 1+0 (RAID 10).

The following table compares these striping options.

Configuration	Use	
RAID 0	Backup scenarios with databases that have data replication.	
RAID 1+0 (RAID 10)	Data loss isn't operationally tolerable. Your application or databases can't tolerate loss of a volume or instance.	

Average Queue Length

The queue length is the number of pending I/O requests. To benefit from Provisioned IOPS, a Provisioned IOPS volume must maintain an average queue length (rounded up to the nearest whole number) of 1 for every 200 provisioned IOPS in a minute. For a Provisioned IOPS volume of 500, the queue length average must be 3. If the average queue length is less than 3 for this volume, you aren't consistently sending enough I/O requests.

Prepare a New Volume

There is a 5 to 50 percent reduction in IOPS when you first access the data on a volume. Performance is restored after the data is accessed once. Therefore, we recommend that you read or write to all the blocks on your volume before you use it.

In Linux, the following command reads from all blocks on a volume.

```
$ dd if=/dev/md0 of=/dev/null
```

Achieve Consistent Performance

To achieve consistent performance from your EBS volumes:

* Use EBS-Optimized instances
* Plan for snapshots

Use EBS-Optimized Instances

The following table provides examples of when you'd want to use an EBS-Optimized instance with each type of EBS volume.

EBS-Optimized instance	Volume type	Result
Yes	Provisioned IOPS	

EBS-Optimized instance	Volume type	
No	Provisioned IOPS	
Yes	Standard EBS	

EBS-Optimized instance	Volume type	[illegible]
No	Standard EBS	[illegible]

You must use EBS-Optimized instances to get the full performance benefits of Amazon EBS Provisioned IOPS volumes. For more information, see EBS-Optimized Instances (p. 88).

Plan for Snapshots

Frequent snapshots provide a higher level of data durability, but they also degrade the performance of your application while the snapshot is in progress. This trade off becomes critical when you have data that's changing rapidly.

To minimize the impact of snapshots on performance of a master node, create snapshots from a read replica of your data. Ideally, create these snapshots during off-peak usage.

Monitor IOPS Performance

Provisioned IOPS volumes deliver within 10 percent of the provisioned IOPS performance 99.9 percent of the time over a given year.

These are the key indicators of IOPS performance:

- The total throughput
- The average queue length

To ensure the consistency of Provisioned IOPS, an application must send a sufficient number of I/O requests. If an application does so, the EBS volume maintains a queue length that ensures IOPS performance. The volumes must maintain an average queue length of 1 (rounded up to the nearest whole

number) for every 200 provisioned IOPS in a minute. For a Provisioned IOPS volume of 500 IOPS, the volume must maintain an average queue length of 3.

The following are possible reasons why a volume isn't meeting the performance threshold:

- The application isn't sending a sufficient number of I/O requests.
- The application is sending I/O requests that have a block size larger than 16 KB.
- [Striped volumes] The application isn't sending I/O requests randomly and consistently across all volumes in the stripe set. Every volume in the stripe set must maintain the recommended average queue length.
- Snapshots are being created frequently during peak usage.

Benchmark Provisioned IOPS Volumes

This section demonstrates how you can test the performance of Provisioned IOPS volumes by simulating workloads similar to those of a database application. The process is as follows:

- Launch an EBS-Optimized instance
- Create Provisioned IOPS volumes
- Attach the volumes to your EBS-Optimized instance, then format and mount them
- Create a RAID array from the volumes
- Install a tool to benchmark I/O performance
- Benchmark the I/O performance of your volumes
- Delete your volumes and terminate your instance so that you don't continue to incur charges

Set Up Your Instance

To get optimal performance from a Provisioned IOPS volume, we recommend that you use an EBS-Optimized instance. EBS-Optimized instances deliver dedicated throughput between Amazon EC2 and Amazon EBS, with options between 500 and 1000 Mbps, depending on the instance type.

To create an EBS-Optimized instance, select **Launch as an EBS-Optimized instance** when launching the instance using the EC2 console, or specify **--ebs-optimized** when using the command line. Be sure that you launch one of the instance types that supports this option. For the example tests in this topic, we recommend that you launch an m1.xlarge instance. For more information, see EBS-Optimized Instances (p. 88).

To create a Provisioned IOPS volume, select **Provisioned IOPS (io1)** when creating the volume using the EC2 console, or specify **--type io1 --iops**_iops_ when using the command line. For information about attaching these volumes to your instance, see Attaching a Volume to an Instance (p. 450).

For the example tests, we recommend that you create a RAID array with 6 volumes, which offers a high level of performance. Because you are charged by the GB used and the number of provisioned IOPS, not the number of volumes, there is no additional cost for creating multiple, smaller volumes and using them to create a stripe set. If you're using Oracle ORION to benchmark your volumes, it can simulate striping the same way that Oracle ASM does, so we recommend that you let ORION do the striping. If you are using a different benchmarking tool, you'll need to stripe the volumes yourself.

For information about creating a striped volume on Windows, see Create a Striped Volume in Windows.

To create a six-volume stripe set on Linux, use a command like the following.

```
$ sudo mdadm --create /dev/md0 --level=0 --chunk=64 --raid-devices=6 /dev/sdf
/dev/sdg /dev/sdh /dev/sdi /dev/sdj /dev/sdk
```

For this example, the file system we use is XFS. You should use the file system that meets your requirements. On Amazon Linux, use the following command to install the XFS file system.

```
$ sudo yum install -y xfsprogs
```

Then, use the following commands to create and mount the XFS file system.

```
$ sudo mkdir -p /media/p_iops_vol0 && sudo mkfs.xfs /dev/md0 &&
sudo mount -t xfs /dev/md0 /media/p_iops_vol0 && sudo chown ec2-user:ec2-user
/media/p_iops_vol0/
```

Finally, to get the most accurate results while running these tests, you must pre-warm the volume. For example, on Linux you can read each block on the volume using the following command.

```
$ dd if=/dev/md0 of=/dev/null
```

On Windows, formatting the volume pre-warms it.

Important

Unless you pre-warm the volume, you might see between a 5 to 50 percent reduction in IOPS when you first access it.

Install Benchmark Tools

The following are among the possible tools you can install and use to benchmark the performance of a Provisioned IOPS volume.

Tool	Platform	Description
fio	Linux, Windows	For benchmarking I/O performance. (Note that you need libaio.)

Tool	Platform	Description
Oracle ORION	Linux, Windows	Calibrate the I/O performance of storage systems to be used with Oracle databases.
SQLIO	Windows	Determine the I/O capacity of a storage configuration to be used with MS SQL Server.

Example Benchmarking Commands

These benchmarking tools support a wide variety of test parameters. You should use commands that approximate the workloads your volumes will support. These commands are intended as examples to help you get started.

Run the following commands on an EBS-Optimized instance with attached Provisioned IOPS volumes that have been pre-warmed.

When you are finished testing your volumes, see these topics for help cleaning up: Deleting an Amazon EBS Volume (p. 474) and Terminating Instances (p. 314).

fio Commands

Run fio on the stripe set that you created.

By default, **fio** is installed in /user/local/bin, which isn't in root's path. You should update your path, or use **chmod** to enable **fio** to be run as ec2-user.

The following command performs 16 KB random write operations.

```
fio --directory=/media/p_iops_vol0
--name fio_test_file --direct=1 --rw=randwrite --bs=16k --size=1G
--numjobs=16 --time_based --runtime=180 --group_reporting
```

The following command performs 16 KB random read operations.

```
fio --directory=/media/p_iops_vol0
--name fio_test_file --direct=1 --rw=randread --bs=16k --size=1G
--numjobs=16 --time_based --runtime=180 --group_reporting
```

For more information about interpreting the results, go to this Linux tutorial: Inspecting disk IO performance with fio.

Oracle ORION Commands

Run ORION on the Provisioned IOPS volumes, having it simulate Oracle ASM striping instead of providing it with a stripe set that uses Windows striping.

In the directory where you installed ORION, create a file, `piops_test.lun`, to specify the volumes for your stripe set. The following example file specifies six Provisioned IOPS volumes to be striped.

```
\\.\D:
\\.\E:
\\.\F:
\\.\G:
\\.\H:
\\.\I:
```

The following command performs 16 KB random I/O operations (80 percent reads and 20 percent writes), simulating 64 KB RAID-0 stripes.

```
orion -run advanced -testname piops_test -size_small 16 -size_large 16
-type rand -simulate raid0 -stripe 64 -write 80 -matrix detailed -num_disks 6
```

After the command is finished, ORION generates output files with the results in the same directory. For more information about ORION, go to its User Guide.

SQLIO Commands

Run SQLIO on the stripe set that you created.

Create a file, `param.txt`, to specify your striped set. The contents of this file should look something like this (here, d:\ corresponds to the striped set, and the test uses 6 threads and a 10 GB file).

```
d:\bigtestfile.dat 6 0x0 10240
```

The following command performs 16 KB random data writes.

```
sqlio -kW -s600 -frandom -t8 -o8 -b16 -LS -BH -Fparam.txt
```

The following command performs 16 KB random data reads.

```
sqlio -kR -s600 -frandom -t8 -o8 -b16 -LS -BH -Fparam.txt
```

The results are displayed in the Command Prompt window. For more information about SQLIO, see the `readme.txt` file in your SQLIO installation directory.

Creating an Amazon EBS Volume

Topics

- AWS Management Console (p. 448)
- Command Line Tools (p. 449)
- API (p. 449)

To use Amazon EBS, you first create a volume that can be attached to any Amazon EC2 instance within the same Availability Zone. You can create an Amazon EBS volume with data from a snapshot stored in Amazon S3 or as an empty volume. You can also create and attach Amazon EBS volumes when you launch instances. For more information on volumes and snapshots, see Amazon Elastic Block Store (Amazon EBS) (p. 434).

New volumes created from existing Amazon S3 snapshots load lazily in the background. This means that after a volume is created from a snapshot, there is no need to wait for all of the data to transfer from Amazon S3 to your Amazon EBS volume before your attached instance can start accessing the volume and all its data. If your instance accesses data that hasn't yet been loaded, the volume immediately downloads the requested data from Amazon S3, and continues loading the rest of the data in the background.

Accessing data for the first time from Amazon S3 might cause latency during the loading period. To avoid the possibility of an increased latency during the background loading of the data, you first access/read all performance-critical data from the volume (or read the entire volume) to ensure it has been loaded to the EBS volume from Amazon S3 before running the application.

AWS Management Console

To create an Amazon EBS volume

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Click **Volumes** in the **Navigation** pane.

   The console displays a list of current volumes.
3. Click **Create Volume**.

   The **Create Volume** dialog box appears.
4. Configure the following settings and click **Yes, Create**.

   - The size of the volume (in GiB)

   - The Availability Zone in which to launch the instance. For more information, see Regions and Availability Zones (p. 5)

   - The ID of the snapshot from which you are launching the volume (optional)

   - The volume type. For more information, see EBS Volume Types (p. 436).

   - [Provisioned IOPS volumes] The input/output operations per second (IOPS) that the volume can support.

Note

If you specify both a volume size and a snapshot ID, the size must be equal to or greater than the snapshot size. Any AWS Marketplace product codes from the snapshot are propagated to the volume.

Note

Any AWS Marketplace product codes from the snapshot are propagated to the volume.

Command Line Tools

To create an Amazon EBS volume

1. Use the ec2-create-volume command to create the volume.

   ```
   PROMPT>  ec2-create-volume --size 80 --availability-zone us-east-1a
   ```

   Amazon EC2 returns information about the volume that is similar to the following example.

   ```
   VOLUME  vol-c7f95aae   80    us-east-1a   creating    2010-03-
   30T13:54:37+0000    standard
   ```

2. To check whether the volume is ready, use the ec2-describe-volumes command.

   ```
   PROMPT>  ec2-describe-volumes volume_id
   ```

   Amazon EC2 returns information about the volume that is similar to the following example.

   ```
   VOLUME  vol-c7f95aae    80    us-east-1a   available   2010-03-
   30T13:54:37+0000    standard
   ```

   Note

   Any AWS Marketplace product codes from the snapshot are propagated to the volume.

API

To create an Amazon EBS volume, use the CreateVolume action. Construct the following request.

```
https://ec2.amazonaws.com/
?Action=CreateVolume
&Size=size
&AvailabilityZone=zone
&AUTHPARAMS
```

The following is an example response.

```
<CreateVolumeResponse xmlns="http://ec2.amazonaws.com/doc/2012-07-20/">
  <requestId>06eabcdf-95b8-424f-87dc-da8e976f7858</requestId>
```

```
        <volumeId>vol-d11fbbb8</volumeId>
        <size>80</size>
        <snapshotId/>
        <availabilityZone>us-east-1a</availabilityZone>
        <status>creating</status>
        <createTime>2010-03-23T09:16:24.000Z</createTime>
        <volumeType>standard</volumeType>
</CreateVolumeResponse>
```

Attaching a Volume to an Instance

Topics

- AWS Management Console (p. 451)
- Command Line Tools (p. 451)
- API (p. 452)

This section describes how to attach a volume that you created to an instance.

The following table lists the available device names on Amazon EC2. You can specify these names when attaching a volume to a running instance or when attaching a volume when launching an instance using a block device mapping. The block device driver for the instance assigns the actual volume names when mounting the volumes, and these names can be different than the names that Amazon EC2 recommends. For more information about the instance store, see Amazon EC2 Instance Store (p. 479). For information about the root device storage, see Root Device Volume (p. 117).

Instance Type	Possible for Connection	Reserved for Root	Instance Store Volumes	Recommended for Connection
Linux / UNIX	/dev/sd[a-z] /dev/sd[a-z][1-15] /dev/hd[a-z] /dev/hd[a-z][1-15]	/dev/sda1	/dev/sd[b-e]	/dev/sd[f-p] /dev/sd[f-p][1-6]
Windows	xvd[a-p] /dev/sda[1-2] /dev/sd[b-e]	/dev/sda1	xvd[a-e]	xvd[f-p]

Important

For Linux/UNIX instance types, we've received reports that some custom kernels might have restrictions that limit use to /dev/sd[f-p] or /dev/sd[f-p][1-6]. If you're having trouble using /dev/sd[q-z] or /dev/sd[q-z][1-6], try switching to /dev/sd[f-p] or /dev/sd[f-p][1-6].

After you have mapped a device using one of these device names, it is mapped to either a scsi drive (sd[a-z]) or an xvd drive (xvd[a-p]), depending on the block device driver of your kernel. If the kernel has an xvd driver, the device is mapped to xvd[a-p]. If the kernel doesn't have an xvd driver, the device is mapped to sd[a-z][1-15].

Base Windows and CentOS HPC (Cluster Compute) images come bundled with an xvd driver, so the devices on Windows and CentOS HPC (Cluster Compute) instances are mapped to xvd[a-p]. The xvd drivers (xvd[a-p]) don't support the use of trailing numbers (xvd[a-p][1-15]). We recommend that you use device names to xvd[a-p] on instances that have xvd drivers.

Depending on the size of your instance, Amazon EC2 provides instance store volumes on sd[b-e] (on Linux/UNIX) or xvd[a-e] (on Windows and CentOS HPC). Although you can connect your Amazon EBS volumes using these device names, we highly recommend that you don't because the behavior can be unpredictable.

An Amazon EC2 Windows AMI comes with an additional service installed, the **Ec2Config Service**. The Ec2Config Service runs as a local system and performs various functions to prepare an instance when it first boots up. After the devices have been mapped with the drives, the Ec2Config Service then initializes and mounts the drives. The root drive is initialized and mounted as `c:\`. The instance stores that comes attached to the instance are initialized and mounted as `d:\`, `e:\`, etc. By default, when an EBS volume is attached to a Windows instance, it may show up as any drive letter on the instance. You can change the settings of the Ec2Config Service to set the drive letters of the EBS volumes per your specifications. For more information, go to Using Ec2Config Service in the *Microsoft Windows Guide*.

AWS Management Console

To attach an Amazon EBS volume to an instance

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Click **Volumes** in the **Navigation** pane.
    The console displays a list of current volumes.
3. Select a volume and click **Attach Volume**.
    The **Attach Volume** dialog box appears.
4. Select the instance to attach the volume to from the **Instance** list box (only instances in the same Availability Zone as the volume are displayed).
5. Select how the device is exposed to the instance from the **Device** list box.
6. Click **Attach** to attach the volume to the instance.

    Note

    The volume and instance must be in the same Availability Zone.

    Note

    You are limited to total of 16 EBS volume attachments on a Windows instance.

    Note

    If a volume has an AWS Marketplace product code:

    *   The volume can only be attached to the root device of a stopped instance.
    *   You must be subscribed to the AWS Marketplace code that is on the volume.
    *   The configuration (instance type, operating system) of the instance must support that specific AWS Marketplace code. For example, you cannot take a volume from a Windows instance and attach it to a Linux instance.
    *   AWS Marketplace product codes are copied from the volume to the instance.

For an overview of the AWS Marketplace, go to https://aws.amazon.com/marketplace/help/200900000. For details on how to use the AWS Marketplace, see AWS Marketplace.

Command Line Tools

To attach an Amazon EBS volume to an instance, use ec2-attach-volume.

```
PROMPT>   ec2-attach-volume volume_id -i instance_id -d device
```

Amazon EC2 returns information using this syntax.

```
ATTACHMENT volume_id instance_id device attaching date_time
```

This example attaches volume `vol-4d826724` to `instance i-6058a509` in Linux and UNIX and exposes it as device `/dev/sdh`.

```
PROMPT>   ec2-attach-volume vol-4d826724 -i i-6058a509 -d /dev/sdh
```

```
ATTACHMENT vol-4d826724 i-6058a509 /dev/sdh attaching 2010-03-30T13:58:58+0000
```

This example attaches volume `vol-4d826724` to `instance i-6058a509` in Windows and exposes it as device `xvdf`.

```
PROMPT>   ec2-attach-volume vol-4d826724 -i i-6058a509 -d xvdf
```

```
ATTACHMENT vol-4d826724 i-6058a509 xvdf attaching 2010-03-30T13:58:58+0000
```

Note

The volume and instance must be in the same Availability Zone.

Note

You are limited to total of 16 EBS volume attachments on a Windows instance.

Note

If a volume has an AWS Marketplace product code:

- The volume can only be attached to the root device of a stopped instance.
- You must be subscribed to the AWS Marketplace code that is on the volume.
- The configuration (instance type, operating system) of the instance must support that specific AWS Marketplace code. For example, you cannot take a volume from a Windows instance and attach it to a Linux instance.
- AWS Marketplace product codes are copied from the volume to the instance.

For an overview of the AWS Marketplace, go to https://aws.amazon.com/marketplace/help/200900000. For details on how to use the AWS Marketplace, see AWS Marketplace.

API

To attach an Amazon EBS volume to an instance, use AttachVolume. Construct the following request.

```
https://ec2.amazonaws.com/
?Action=AttachVolume
&VolumeId=volume-id
```

```
&InstanceId=instance-id
&Device=device
&AUTHPARAMS
```

The following is an example response.

```
<AttachVolumeResponse xmlns="http://ec2.amazonaws.com/doc/2012-07-20/">
  <requestId>59dbff89-35bd-4eac-99ed-be587EXAMPLE</requestId>
  <volumeId>vol-4d826724</volumeId>
  <instanceId>i-6058a509</instanceId>
  <device>/dev/sdh</device>
  <status>attaching</status>
  <attachTime>2008-05-07T11:51:50.000Z</attachTime>
</AttachVolumeResponse>
```

Note

The volume and instance must be in the same Availability Zone.

Note

You are limited to total of 16 EBS volume attachments on a Windows instance.

Note

If a volume has an AWS Marketplace product code:

- The volume can only be attached to the root device of a stopped instance.
- You must be subscribed to the AWS Marketplace code that is on the volume.
- The configuration (instance type, operating system) of the instance must support that specific AWS Marketplace code. For example, you cannot take a volume from a Windows instance and attach it to a Linux instance.
- AWS Marketplace product codes are copied from the volume to the instance.

For an overview of the AWS Marketplace, go to https://aws.amazon.com/marketplace/help/200900000. For details on how to use the AWS Marketplace, see AWS Marketplace.

Describing Volumes

Topics

You can list information about a volume, including the specific instance the volume is attached to.

AWS Management Console

To view information about an Amazon EBS volume

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Click **Volumes** in the **Navigation** pane.

The console displays a list of current volumes and the instances they are attached to.

3. To view more information about a volume, select it.

   Information about the volume appears in the lower pane.

Command Line Tools

To describe volumes and list information about the volumes that you own, use the ec2-describe-volumes command.

```
PROMPT>  ec2-describe-volumes
```

Amazon EC2 returns information similar to the following.

```
VOLUME vol-4d826724 80          us-east-1a in-use    2010-03-30T13:58:58+0000
    standard
ATTACHMENT vol-4d826724 i-6058a509 /dev/sdh attached  2010-03-30T13:54:55+0000
VOLUME vol-50957039 13          us-east-1a available  2010-03-24T08:01:44+0000
    standard
VOLUME vol-6682670f 1           us-east-1a in-use     2010-03-30T08:11:01+0000
    standard
ATTACHMENT vol-6682670f i-69a54000 /dev/sdh attached  2010-03-30T09:21:14+0000
```

This information includes the volume ID, capacity, status (in-use or available), and creation time of each volume. If the volume is attached, an attachment line shows the volume ID, the instance ID to which the volume is attached, the device name exposed to the instance, its status (attaching, attached, detaching, detached), and when it attached.

> **Tip**
>
> You can use **ec2-describe-volumes** to filter the results to only the volumes that match the criteria you specify.

To describe instances and list volumes that are attached to running instances, use the ec2-describe-instances command.

```
PROMPT>  ec2-describe-instances
```

Amazon EC2 returns information similar to the following.

```
RESERVATION     r-f25e6f9a      111122223333    default
INSTANCE        i-84b435de      ami-b232d0db    ec2-184-73-201-68.compute-
1.amazonaws.comdomU-12-31-39-00-86-35.compute-1.internal    running gsg-keypair
    0    m1.small 2010-03-30T08:43:48+0000    us-east-1a    aki-94c527fd
ari-96c527ff    monitoring-disabled    184.73.201.68   10.254.137.191     ebs
BLOCKDEVICE     /dev/sda1       vol-cf13b3a6    2010-03-30T08:01:44.000Z
true
BLOCKDEVICE     /dev/sdh        vol-c7f95aae    2010-03-30T13:58:58.000Z
true
```

Tip

You can use **ec2-describe-instances** to filter the results to only the instances that match the criteria you specify.

For more information about block device mapping, see Block Device Mapping (p. 488).

API

To describe volumes and list information about all volumes that you own, use the DescribeVolumes action. Construct the following request.

```
https://ec2.amazonaws.com/
?Action=DescribeVolumes
&AUTHPARAMS
```

The following is an example response.

```
<DescribeVolumesResponse xmlns="http://ec2.amazonaws.com/doc/2012-07-20/">
<volumeSet>
  <item>
    <volumeId>vol-4282672b</volumeId>
    <size>80</size>
    <status>in-use</status>
    <createTime>2008-05-07T11:51:50.000Z</createTime>
    <attachmentSet>
      <item>
        <volumeId>vol-4282672b</volumeId>
        <instanceId>i-6058a509</instanceId>
        <size>80</size>
        <snapshotId>snap-12345678</snapshotId>
        <availabilityZone>us-east-1a</availabilityZone>
        <status>attached</status>
        <attachTime>2008-05-07T12:51:50.000Z</attachTime>
      </item>
    </attachmentSet>
  </item>
  ...
</volumeSet>
```

Tip

You can use `DescribeVolumes` to filter the results to only the volumes that match the criteria you specify.

Making an Amazon EBS Volume Available for Use

Topics
- Make the Volume Available on Linux (p. 456)
- Make the Volume Available on Windows (p. 456)

Inside the instance, an Amazon EBS volume is exposed as a normal block device and can be formatted as any file system and mounted.

After making the Amazon EBS volume available for use, you can take snapshots of it for backup purposes or to use as baselines to launch new volumes.

Make the Volume Available on Linux

To create an ext3 file system on the Amazon EBS volume and mount it as /mnt/data-store

1. Enter the following command to create the file system.

   Caution

   This step assumes that you're mounting an empty volume. If you're mounting a volume that already has data on it (e.g., a public dataset), don't use `mkfs` before mounting the volume (skip to the next step instead). Otherwise you'll format the volume and delete the existing data.

   ```
   $   sudo mkfs -t ext3 /dev/sdh
   ```

2. Enter the following command to create the directory.

   ```
   $   sudo mkdir /mnt/data-store
   ```

3. Enter the following command to mount the volume.

   ```
   $   sudo mount /dev/sdh /mnt/data-store
   ```

Any data written to this file system is written to the Amazon EBS volume and is transparent to applications using the device.

   Note

   To enable the instance to reconnect to an Amazon EBS volume on reboot, add the device to the fstab or create a script that automatically mounts the volume on start-up.

Make the Volume Available on Windows

To use an Amazon EBS volume

1. Log in to your instance using Remote Desktop.

2. On the taskbar, click **Start**, and then click **Run**.

3. Type **diskmgmt.msc** and click **OK**. The Disk Management utility opens.

   Caution

   If you're mounting a volume that already has data on it (for example, a public dataset), make sure you don't reformat the volume and delete the existing data.

4. Right-click the new Amazon EBS volume, select **Online**.

Your new EBS volume is now available for use. Any data written to this file system is written to the Amazon EBS volume and is transparent to applications using the device.

Monitoring the Status of Your Volumes

Volume status provides information about the I/O (read/write) capability of your Amazon Elastic Block Store (Amazon EBS) volume. It is designed to provide you the information you need in order to understand when your EBS volumes are impaired and what actions to consider taking. As with instance status checks, the volume status checks return a pass or a fail status. If all checks pass, the overall status of the volume is OK. If the check fails, the overall status is impaired.

Volume status checks let you know when an EBS volume's data is potentially inconsistent. By default, AWS disables I/O access to the volume from any attached EC2 instance to prevent undetectable, latent data corruption. At that point, the volume status check fails showing the volume status is impaired. In addition, volume status shows an event letting you know when the I/O became disabled and what action you may take to resolve the impaired status of the volume, which is to enable the volume's I/O.

AWS waits until you tell us you are aware of the potential data inconsistency and are ready to enable the volume's I/O. This lets you choose if and when to enable the volume's I/O and provides you the opportunity to decide whether to simply continue application use of the volume or to check its data consistency first by running a consistency check. Example volume consistency checking tools you could use are fsck (File System Check) when running Linux and chkdsk (Check Disk) when running Windows.

Alternatively, you can set an EBS volume attribute to automatically enable I/O operations on a potentially inconsistent volume. In this case, the volume status check will continue to pass, and an event will be created showing when the volume was determined to be potentially inconsistent but had its I/O automatically enabled. You might choose to do this when availability of the volume is more important than data consistency. For example, in the case of a boot volume only written to for logging, you may want immediate availability of the volume to optimize for availability of the instance before checking the volume's consistency or replacing it later.

Note

Volume status only has one status check. It does not check volume state as reported by DescribeVolumes. Therefore, it does not detect volumes in the `ERROR` state (i.e., when a volume is incapable of accepting I/Os because it is in an error state.)

Topics

Monitoring Volumes with Status Checks

You can view the results of volume status checks to identify any impaired volumes and take any necessary actions.

Viewing Status

AWS provides you with several ways to view and work with status checks: You can use the AWS Management Console, interact directly with the API, or use the command line interface.

AWS Management Console

To view status checks

1. Sign in to the AWS Management Console and open the Amazon EC2 console at
    https://console.aws.amazon.com/ec2/.

2. In the **Navigation** pane, under **ELASTIC BLOCK STORE**, click **Volumes**.

   Note

   You can also use the **Events** pane to view all events for your instances and volumes in a single pane.

3. On the **EBS Volumes** page, the **Status Checks** column lists the operational status of each volume.

4. To view an individual volume's status, select the volume, and then click the **Status Checks** tab.

   Note

   If you have a volume with a failed status check (status shows as impaired), see Working with an Impaired Volume (p. 461).

Using the Command Line

This following examples show how to use the command line to describe the status of your volumes.

Run this command	To do this
`ec2-describe-volume-status`	Return the status of all volumes in a paged format
`ec2-describe-volume-status --filter "volume-status.status=impaired"`	Returns the status of all volumes with a volume status of impaired

For more information about using the `ec2-describe-volume-status` command, see ec2-describe-volume-status in the *Amazon Elastic Compute Cloud Command Reference Guide.*

> **Note**
>
> If you have a volume with a failed status check (status shows as impaired), see Working with an Impaired Volume (p. 461).

Using the API

You can use the `DescribeVolumeStatus` action to retrieve the status of your volumes. For more information, see DescribeVolumeStatus in the *Amazon Elastic Compute Cloud API Reference Guide*.

Monitoring Volume Events

When EBS determines a volume's data to be potentially inconsistent and disables its I/O causing the status check to fail, a volume status event shows the cause of the status check failure. Each event shows a start time indicating the time at which the event occurred and a duration field indicating the time that I/O operation was disabled for the volume. The end time is populated when the volume's I/O operations are enabled.

Volume status events will show one of the following descriptions:

- **Awaiting Action: Enable IO**—Volume data is potentially inconsistent. I/O operations are disabled until you enable them.
- **IO Enabled**—I/O operations have been enabled on this volume. The event description changes to `IO Enabled` after you explicitly enable I/O to the volume. AWS recommends checking for data inconsistencies after enabling I/O before continuing application use of the data. For more information, see Working with an Impaired Volume (p. 461).
- **IO Auto-Enabled**—I/O operations were automatically enabled on this volume after an event occurred based on the Auto-Enable IO setting on the volume.

> **Note**
>
> As a precaution, after a service disruption event when a volume is brought back online, I/O operations are disabled for any potentially inconsistent volumes. If you want to automatically allow I/O operations on a volume with potential data inconsistencies, you can change the `Auto-Enable IO` volume attribute. For more information about how and when to change the default setting, see Working with an Impaired Volume (p. 461).
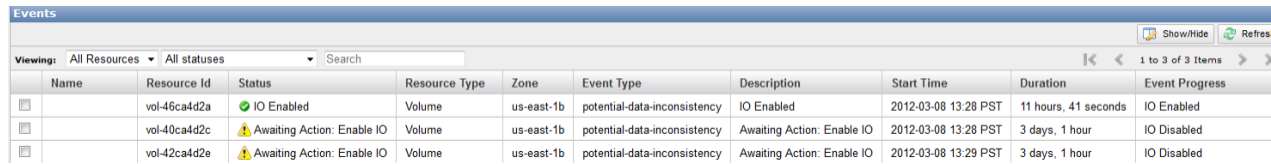
Monitoring Volume Events with Volume Status

You can view events for your volumes using the AWS Management Console, the API, or the command line interface.

AWS Management Console

To view events for your volumes

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the **Navigation** pane, click **Events**. You see a list of all instances and volumes and any associated events.
3. On the **Events** page, in **Viewing**, select **Volumes** to view only volume status. You can also filter on specific status types.

4. To view the event details of a specific volume, select the volume in the **Events** page.

Using the Command Line

To view events for your volumes using the command line

• Enter the following command:

```
PROMPT> ec2-describe-volume-status
```

Amazon EC2 returns output similar to the following:

```
PROMPT> ec2-describe-volume-status  vol-111111, vol-222222
Type          VolumeId    AvailabilityZone VolumeStatus
VOLUME        vol-111111 us-east-1a        ok
VOLUME        vol-222222 us-east-1b        impaired
Type          Name        Status
VOLUMESTATUS  io-enabled failed
Type  EventType       NotBefore              NotAfter EventId   EventDe
scription
EVENT potential-data-inconsistency 2011-12-01T14:00:00.000Z      848721011
  This is an example
Type   ActionCode               EventId    EventType
      EventDescription
ACTION enable-volume-io         848721011  potential-data-inconsistency
  This is an example
```

For more information about using the `ec2-describe-volume-status` command, see
ec2-describe-volume-status in the *Amazon Elastic Compute Cloud Command Reference Guide*.

Note

If you have a volume with a failed status check, see Working with an Impaired Volume (p. 461).

Using the API

You can use the DescribeVolumeStatus action to retrieve the status of your volumes and any associated events. For more information, see DescribeVolumeStatus in the *Amazon Elastic Compute Cloud API Reference Guide*.

Working with an Impaired Volume

This section discusses the options available if a volume is impaired because the volume's data is potentially inconsistent. You can do any of the following:

- Run a consistency check in place: Enable I/O and then perform a data consistency check on the volume using its attached instance.
- Run a consistency check using another instance: Force detach the volume, enable I/O, attach the volume to another instance, and then perform a data consistency check. (Note that force detaching a volume may lose in-progress write I/Os that were suspended when I/O was disabled.)
- Delete the volume if you no longer need it (e.g., you are running a database and would rather revert to a known good database state via a snapshot backup copy).

Option 1: Perform a Consistency Check on the Volume Attached to its Instance

The simplest option is to perform a consistency check on the volume while the volume is still attached to its Amazon EC2 instance.

To perform a consistency check on an attached volume

1. Stop any applications from using the volume.
2. Enable I/O on the volume.

   Note

   To use the command line interface to enable I/O operations, see ec2-enable-volume-io in the *Amazon Elastic Compute Cloud Command Line Reference Guide*. To use the API, see EnableVolumeIO in the *Amazon Elastic Compute Cloud API Reference Guide*.

   a. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
   b. In the **Navigation** pane, under **ELASTIC BLOCK STORE**, click **Volumes**.
   c. Select the volume on which you want to enable I/O operations.
   d. On the **Details** tab, click **Enable Volume IO**.

    e.   In **Enable Volume IO**, click **Yes, Enable**.

3.   Check the data on the volume.

    a.   (Optional) Run fsck (file system check) on Linux or chkdsk (check disk) on Windows.

    b.   Review application logs.

    c.   (Optional) If the instance has been impaired for more than 20 minutes you can contact support. Click **Troubleshoot Now**, and then on the **Troubleshoot Status Checks** dialog box, click **Contact Support** to submit a support case.

Option 2: Perform a Consistency Check on the Volume in Isolation

Use the following procedure to isolate the volume from a production environment.

To perform a consistency check on a volume in isolation

1.   Stop any applications from using the volume.

2.   Force detach the volume from the instance.

> **Note**
>
> Force detach may cause the loss of write I/Os that were suspended when volume I/O was disabled.

    a.   Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

    b.   In the **Navigation** pane, under **ELASTIC BLOCK STORE**, click **Volumes**.

    c.   Select the volume that you want to force detach.

    d.   Click **More**, and then click **Force Detach**.

    e.   In the **Force Detach Volume** dialog box, click **Yes, Force**.

3.   Enable I/O on the volume.

> **Note**
>
> To use the command line interface to enable I/O operations, see ec2-enable-volume-io in the *Amazon Elastic Compute Cloud Command Line Reference Guide*. To use the API, see EnableVolumeIO in the *Amazon Elastic Compute Cloud API Reference Guide*.

    a.   Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

    b.   In the **Navigation** pane, under **ELASTIC BLOCK STORE**, click **Volumes**.

    c.   Select the volume on which you want to enable I/O operations.

    d.   On the **Details** tab, click **Enable Volume IO**.

```
1 Volume selected

  Volume: vol-2670f74a

                                IO operations have been disabled since 104 days, 12 hours ago. Data inconsistencies may e

  Details    Status Checks    Monitoring    Tags

  Volume ID:        vol-2670f74a              Alarm Status:

  Capacity:         10 GiB                    Snapshot:

  Created:          2012-03-27 18:53 GMT+0200 Zone:

  Status:           available                 Attachment:

  Volume Type:      standard                  IOPS:
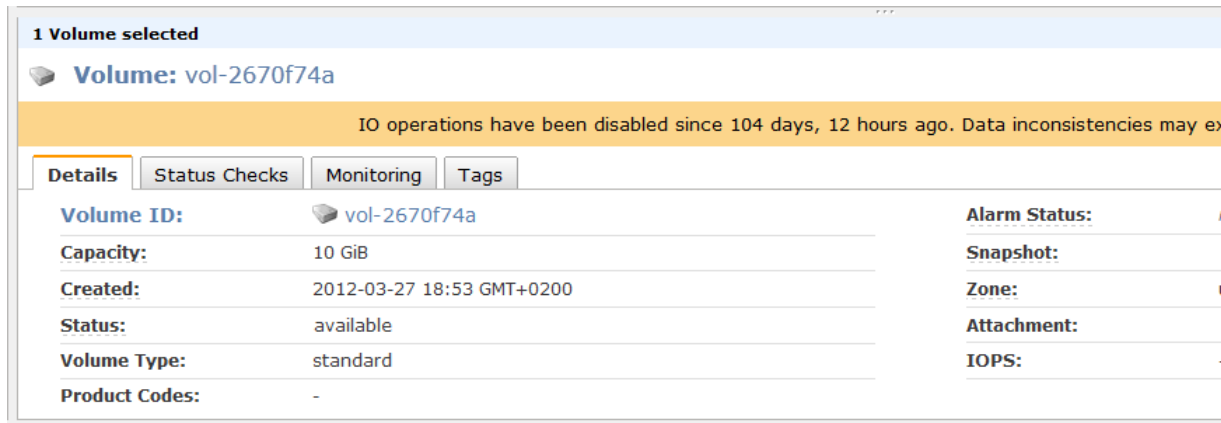
  Product Codes:    -
```

e. In **Enable Volume IO**, click **Yes, Enable**.

4. Attach the volume to another instance.
5. Check the data on the volume.

a. (Optional) Run fsck (file system check) on Linux or chkdsk (check disk) on Windows.
b. Review application logs.
c. (Optional) If the instance has been impaired for more than 20 minutes you can contact support. Click **Troubleshoot Now**, and then on the **Troubleshoot Status Checks** dialog box, click **Contact Support** to submit a support case.

Option 3: Delete the Volume

If you want to remove the volume from your environment, simply delete it. For more information, see Deleting an Amazon EBS Volume (p. 474).

Working with the Volume Auto-Enable IO Setting

This section explains how to view and modify the Auto-Enable setting on a volume.

Viewing the Volume Auto-Enable IO Setting

You can view the Auto-Enable IO setting of a volume using the AWS Management Console, the command line interface, or the API.

AWS Management Console

To view the Auto-Enable IO setting on a volume

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the **Navigation** pane, under **ELASTIC BLOCK STORE**, click **Volumes**.
3. Select the volume on which you want to view the Auto-Enable IO setting.
4. In the lower pane, click the **Status Checks** tab.
5. In **Auto-Enable Volume IO**, view the current setting for your volume.

Command Line Tools

To view the Auto-Enable IO setting on a volume

- Enter the following command to return the volume's Auto-Enable setting:

```
PROMPT> ec2-describe-volume-attribute vol-999999 --auto-enable-io
```

For more information about using the `ec2-describe-volume-attribute` command, see `ec2-describe-volume-attribute` in the *Amazon Elastic Compute Cloud Command Reference Guide*.

API

You can use the DescribeVolumeAttribute action to view the Auto-Enable IO setting on a volume. For more information, see DescribeVolumeAttribute in the *Amazon Elastic Compute Cloud API Reference Guide*.
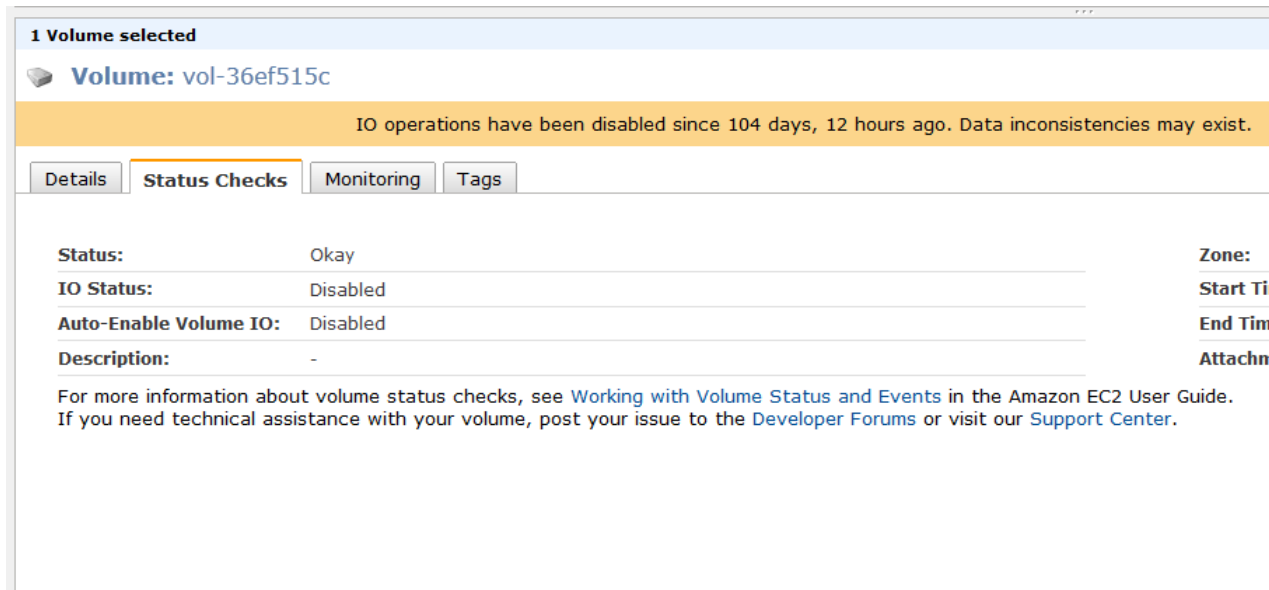
Modifying the Volume Auto-Enable IO Setting

You can modify the Auto-Enable IO setting of a volume using the AWS Management Console, the command line tools, or the API.

AWS Management Console

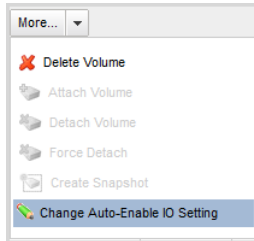**To modify the Auto-Enable IO setting on a volume**

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the **Navigation** pane, under **ELASTIC BLOCK STORE**, click **Volumes**.
3. Select the volume on which you want to modify the attribute.
4. At the top of the Volumes page, click **More**.

Note

>   You can also use the **Auto-Enable IO Setting** button on the Status Checks tab to change
>   this setting.

5.   Click **Change Auto-Enable IO Setting**.

6.   In **Modify Auto-Enable Volume IO**, click the **Auto-Enable Volume IO** check box to automatically
     enable I/O operations to a volume. Clear the check box to disable the setting.

7.   Click **Save**.

Command Line Tools

To modify the Auto-Enable IO setting on a volume

*   Enter the following command to modify the volume's Auto-Enable IO setting:

```
PROMPT> ec2-modify-volume-attribute vol-999999  --auto-enable-io true
```

For more information about using the `ec2-modify-volume-attribute` command, see
ec2-modify-volume-attribute in the *Amazon Elastic Compute Cloud Command Reference Guide*.

API

You can use the ModifyVolumeAttribute action to modify the Auto-Enable IO setting on a volume. For
more information, see ModifyVolumeAttribute in the *Amazon Elastic Compute Cloud API Reference
Guide*.

Creating an Amazon EBS Snapshot

Topics

After writing data to an Amazon EBS volume, you can periodically create a snapshot of the volume to use as a baseline for new volumes or for data backup. If you make periodic snapshots of a volume, the snapshots are incremental so that only the blocks on the device that have changed since your last snapshot are saved in the new snapshot. Even though snapshots are saved incrementally, the snapshot deletion process is designed so that you need to retain only the most recent snapshot in order to restore the volume.

Snapshots occur asynchronously and the status of the snapshot is `pending` until the snapshot is complete.

By default, only you can launch volumes from snapshots that you own. However, you can choose to share your snapshots with specific AWS accounts or make them public. For more information, see Modifying Snapshot Permissions (p. 468).

When a snapshot is created, any AWS Marketplace product codes from the volume are propagated to the snapshot.

AWS Management Console

To create a snapshot

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Click **Snapshots** in the **Navigation** pane.
   The console displays a list of current snapshots.
3. Click **Create Snapshot**.
   The Create Snapshot dialog box appears.
4. Select the volume to create a snapshot for and click **Create**.
   Amazon EC2 begins creating the snapshot.

Command Line Tools

To create a snapshot, use the ec2-create-snapshot command.

```
PROMPT>  ec2-create-snapshot volume_id
```

Amazon EC2 returns information similar to the following example.

```
SNAPSHOT      snap-88fe11e0   vol-c7f95aae   pending 2010-03-30T14:10:38+0000
        111122223333     80
```

When the snapshot is complete, its status changes to `completed`.

API

To create a snapshot, use the CreateSnapshot action. Construct the following request.

```
https://ec2.amazonaws.com/
?Action=CreateSnapshot
&VolumeId=volume-id
&AUTHPARAMS
```

The following is an example response.

```
<CreateSnapshotResponse xmlns="http://ec2.amazonaws.com/doc/2012-07-20/">
  <requestId>fe0d60a3-b804-484e-b558-92518bebe7af</requestId>
  <snapshotId>snap-05b4aa6c</snapshotId>
  <volumeId>vol-d11fbbb8</volumeId>
  <status>pending</status>
  <startTime>2010-03-23T09:43:51.000Z</startTime>
  <progress/>
  <ownerId>999988887777</ownerId>
  <volumeSize>20</volumeSize>
  <description/>
</CreateSnapshotResponse>
```

Describing Snapshots

Topics

This section describes how to view snapshots that you created.

AWS Management Console

To describe snapshots

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Click **Snapshots** in the **Navigation** pane.

   The console displays a list of all snapshots to which you have access and their status.
3. To reduce the list, select an option from the **Viewing** list box. For example, to view only your snapshots, select **Owned by Me**.

   The console displays a new list of snapshots.
4. To view more information about a snapshot, select it.

   Information about the snapshot appears in the lower pane.

Command Line Tools

To describe snapshots, use the ec2-describe-snapshots command.

```
PROMPT>  ec2-describe-snapshots snap-78a54011
```

Amazon EC2 returns information about the snapshot.

```
SNAPSHOT snap-78a54011 vol-4d826724 pending 2008-02-15T09:03:58+0000    60%
```

If the snapshot is in the process of being created, the status is `pending` and a percentage complete is displayed (e.g., 60%). After the snapshot is complete, its status changes to `completed`.

Tip

If you have a large number of snapshots, you can use **ec2-describe-snapshots** to filter the
results to only the snapshots that match the criteria you specify.

API

To describe snapshots, use the DescribeSnapshots action. Construct the following request.

```
https://ec2.amazonaws.com/
?Action=DescribeSnapshots
&SnapshotId=snapshot-id
&AUTHPARAMS
```

The following is an example response.

```
<DescribeSnapshotsResponse xmlns="http://ec2.amazonaws.com/doc/2012-07-20/">
  <requestId>26245bae-2c70-4846-82d3-65784e298820</requestId>
  <snapshotSet>
      <item>
          <snapshotId>snap-05b4aa6c</snapshotId>
          <volumeId>vol-d11fbbb8</volumeId>
          <status>completed</status>
          <startTime>2010-03-23T09:43:52.000Z</startTime>
          <progress>100%</progress>
          <ownerId>999988887777</ownerId>
          <volumeSize>20</volumeSize>
          <description/>
      </item>
  </snapshotSet>
</DescribeSnapshotsResponse>
```

Tip

If you have a large number of snapshots, you can use `DescribeSnapshots` to filter the list to
only the snapshots that match criteria you specify.

Modifying Snapshot Permissions

Topics

This section describes how to modify permissions for your snapshots so that specific AWS accounts or
all Amazon EC2 users can create volumes from them.

Important

When you share a snapshot (whether by sharing it with another AWS account or making it public
to all), you are giving others access to all the data on your snapshot. Share snapshots only with
people with whom you want to share *all* your snapshot data.

AWS Management Console

To modify snapshot permissions

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Click **Snapshots** in the **Navigation** pane.
    The console displays a list of current snapshots and their status.
3. Select a snapshot and click **Permissions**.
    The **Modify Snapshot Permissions** dialog box appears.
4. Choose whether to make the snapshot public or to share it with select AWS accounts:

    > **Important**
    >
    > Making your snapshot public shares all snapshot data with everyone. Snapshots with AWS
    > Marketplace product codes cannot be made public.

    *   To make the snapshot public, select **Public** and click **Save**.
    *   To expose the snapshot only to specific AWS accounts, select **Private**, enter the IDs of those
        AWS accounts, and click **Save**.

    The console modifies permissions for the snapshot.

Command Line Tools

To modify snapshot permissions

1. Use the ec2-describe-snapshot-attribute command to first describe the snapshot's permissions.

    ```
    PROMPT>  ec2-describe-snapshot-attribute snap_id --create-volume-permission
    ```

    If there are no permissions set on the snapshot, the output is empty.
2. Choose whether to make the snapshot public, or to share it with a specific AWS account.

    > **Important**
    >
    > Making your snapshot public shares all snapshot data with everyone. Snapshots with AWS
    > Marketplace product codes cannot be made public.

    *   To make the snapshot public, use the ec2-modify-snapshot-attribute command as follows.

    ```
    PROMPT>  ec2-modify-snapshot-attribute snap_id -c --add all
    ```

    Amazon EC2 returns permission information for the snapshot.

    ```
    createVolumePermission   snap_id   ADD     group   all
    ```

    *   To share the snapshot with a particular AWS account, use the ec2-modify-snapshot-attribute
        command as follows

    ```
    PROMPT>  ec2-modify-snapshot-attribute snap_id -c --add account_id
    ```

Amazon EC2 returns permission information for the snapshot.

```
createVolumePermission   snap_id   ADD        account_id
```

API

To modify snapshot permissions

1. Use the DescribeSnapshotAttribute action to describe the snapshot's permissions. Construct the following request.

    ```
    https://ec2.amazonaws.com/
    ?Action=DescribeSnapshotAttribute
    &SnapshotId=snapshot-id
    &AUTHPARAMS
    ```

    The following is an example response.

    ```
    <DescribeSnapshotAttributeResponse xmlns="http://ec2.amazonaws.com/doc/2012-
    07-20/">
      <requestId>d0d21738-e3da-4077-947d-c9e48472d831</requestId>
      <snapshotId>snap-05b4aa6c</snapshotId>
      <createVolumePermission/>
    </DescribeSnapshotAttributeResponse>
    ```

2. Choose whether to make the snapshot public, or to share it with a specific AWS account.

    Important

    Making your snapshot public shares all snapshot data with everyone. Snapshots with AWS Marketplace product codes cannot be made public.

    *   To make the snapshot public, use the ModifySnapshotAttribute action. Construct the following request.

        ```
        https://ec2.amazonaws.com/
        ?Action=ModifySnapshotAttribute
        &SnapshotId=snapshot-id
        &CreateVolumePermission.Add.1.Group=all
        &AUTHPARAMS
        ```

    *   To share the snapshot with a particular AWS account, use the ModifySnapshotAttribute action as follows.

        ```
        https://ec2.amazonaws.com/
        ?Action=ModifySnapshotAttribute
        &SnapshotId=snapshot-id
        &CreateVolumePermission.Add.1.UserId=111122223333
        &AUTHPARAMS
        ```

Detaching an Amazon EBS Volume from an Instance

Topics

You can detach an Amazon EBS volume from an instance explicitly or by terminating the instance. However, a volume must be unmounted inside the instance before being detached. Failure to do so results in the volume being stuck in the busy state while it is trying to detach, which could possibly damage the file system or the data it contains.

Note

If an Amazon EBS volume is the root device of an instance, it cannot be detached unless the instance is in the stopped state.

If the root volume is detached from an instance with an AWS Marketplace product code, then the AWS Marketplace product codes from that volume are no longer associated with the instance.

This example unmounts the volume and then explicitly detaches it from the instance. This is useful when you want to terminate an instance or attach a volume to a different instance. To verify that the volume is no longer attached to the instance, see Describing Volumes (p. 453).

AWS Management Console

To detach an Amazon EBS volume

1. First, unmount the volume.

   For Linux/UNIX, use the following command to unmount the /dev/sdh device.

   ```
   # umount -d /dev/sdh
   ```

   For Windows, open **Disk Management**, right-click the volume to unmount, and select **Change Drive Letter and Path**. Then, select the mount point to remove and click **Remove**.

2. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

3. Click **Volumes** in the **Navigation** pane.
   The console displays a list of current volumes.

4. Select a volume and click **Detach Volume**.
   A confirmation dialog box appears.

5. Click **Yes, Detach**.
   The volume is detached from the instance.

   Caution

   If your volume stays in the *detaching* state, you can force the detachment using the **Force Detach** button. Forcing the detachment can lead to data loss or a corrupted file system. Use this option only as a last resort to detach a volume from a failed instance. The instance doesn't get an opportunity to flush file system caches or file system metadata. If you use this option, you must perform file system check and repair procedures.

Command Line Tools

To detach an Amazon EBS volume explicitly

1. For Linux/UNIX, use the following command to unmount the /dev/sdh device.

```
# umount -d /dev/sdh
```

   For Windows, open **Disk Management**, right-click the volume to unmount, and select **Change Drive Letter and Path**. Then, select the mount point to remove and click **Remove**.

2. To detach the volume, use the ec2-detach-volume command.

```
PROMPT>  ec2-detach-volume vol-4d826724
```

   Amazon EC2 returns information similar to the following example.

```
ATTACHMENT vol-4d826724 i-6058a509 /dev/sdh detaching 2010-03-30T13:58:58+0000
```

> **Caution**
>
> If your volume stays in the *detaching* state, you can force the detachment using the --force option of **ec2-detach-volume**. Forcing the detachment can lead to data loss or a corrupted file system. Use this option only as a last resort to detach a volume from a failed instance. The instance doesn't get an opportunity to flush file system caches or file system metadata. If you use this option, you must perform file system check and repair procedures.

To detach an Amazon EBS volume by terminating the instance

- Use the ec2-terminate-instances command.

```
PROMPT>  ec2-terminate-instances i-6058a509
```

   Amazon EC2 returns information similar to the following example.

```
INSTANCE    i-6058a509    running shutting-down
```

API

To detach an Amazon EBS volume, use the DetachVolume action. Construct the following request.

```
https://ec2.amazonaws.com/
?Action=DetachVolume
&VolumeId=volume-id
&InstanceId=instance-id
&AUTHPARAMS
```

The following is an example response.

```
<DetachVolumeResponse xmlns="http://ec2.amazonaws.com/doc/2012-07-20/">
  <requestId>59dbff89-35bd-4eac-99ed-be587EXAMPLE</requestId>
  <volumeId>vol-4d826724</volumeId>
  <instanceId>i-6058a509</instanceId>
  <device>/dev/sdh</device>
  <status>detaching</status>
  <attachTime>2008-05-08T11:51:50.000Z</attachTime>
</DetachVolumeResponse>
```

Caution

If your volume stays in the *detaching* state, you can force the detachment using the `Force` option of `DetachVolume`. Forcing the detachment can lead to data loss or a corrupted file system. Use this option only as a last resort to detach a volume from a failed instance. The instance will not have an opportunity to flush file system caches or file system metadata. If you use this option, you must perform file system check and repair procedures.

Deleting an Amazon EBS Snapshot

Topics

This section describes how to delete a snapshot.

Note

- If you make periodic snapshots of a volume, the snapshots are incremental so that only the blocks on the device that have changed since your last snapshot are saved in the new snapshot. Even though snapshots are saved incrementally, the snapshot deletion process is designed so that you need to retain only the most recent snapshot in order to restore the volume.

- You cannot delete a snapshot of the root device of an EBS volume used by a registered AMI. You must first de-register the AMI before you can delete the snapshot.

AWS Management Console

To delete a snapshot

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Click **Snapshots** in the **Navigation** pane.
   The console displays a list of current snapshots.
3. Select a snapshot and click **Delete Snapshot**.
   A confirmation dialog box appears.
4. Click **Yes, Delete**.
   The snapshot is deleted.

Command Line Tools

To delete a snapshot, use the ec2-delete-snapshot command.

```
PROMPT>   ec2-delete-snapshot snapshot_id
```

Amazon EC2 returns information similar to the following example.

```
SNAPSHOT snap-78a54011
```

API

To delete a snapshot, use the DeleteSnapshot action. Construct the following request.

```
https://ec2.amazonaws.com/
?Action=DeleteSnapshot
&SnapshotId=snapshot-id
&AUTHPARAMS
```

The following is an example response.

```
<DeleteSnapshotResponse xmlns="http://ec2.amazonaws.com/doc/2012-07-20/">
  <requestId>59dbff89-35bd-4eac-99ed-be587EXAMPLE</requestId>
  <return>true</return>
</DeleteSnapshotResponse>
```

Deleting an Amazon EBS Volume

Topics

After you no longer need a volume, you can delete it. After deletion, its data is gone and the volume can't be attached to any instance. However, before deletion, you can store a snapshot of the volume, which you can use to recreate the volume later.

This section describes how to delete a volume.

AWS Management Console

To delete a volume

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Click **Volumes** in the **Navigation** pane.
    The console displays a list of current volumes.
3. Select a volume and click **Delete Volume**.
    A confirmation dialog box appears.
4. Click **Yes, Delete**.
    The volume is deleted.

Command Line Tools

To delete a volume, use the ec2-delete-volume command.

```
PROMPT>  ec2-delete-volume vol-4282672b
```

Amazon EC2 returns information similar to the following example.

```
VOLUME vol-4282672b
```

API

To delete a volume, use the DeleteVolume action. Construct the following request.

```
https://ec2.amazonaws.com/
?Action=DeleteVolume
&VolumeId=volume-id
&AUTHPARAMS
```

The following is an example response.

```
<DeleteVolumeResponse xmlns="http://ec2.amazonaws.com/doc/2012-07-20/">
  <requestId>59dbff89-35bd-4eac-99ed-be587EXAMPLE</requestId>
  <return>true</return>
</DeleteVolumeResponse>
```

Using Public Data Sets

Topics

This section describes how to use Amazon EC2 public data sets.

Public Data Set Concepts

Amazon EC2 provides a repository of public data sets that can be seamlessly integrated into AWS cloud-based applications. Amazon stores the data sets at no charge to the community and, like with all AWS services, you pay only for the compute and storage you use for your own applications.

Previously, large data sets such as the mapping of the Human Genome and the US Census data required hours or days to locate, download, customize, and analyze. Now, anyone can access these data sets from an Amazon EC2 instance and start computing on the data within minutes. You can also leverage the entire AWS ecosystem and easily collaborate with other AWS users. For example, you can produce or use prebuilt server images with tools and applications to analyze the data sets. By hosting this important and useful data with cost-efficient services such as Amazon EC2, AWS hopes to provide researchers across a variety of disciplines and industries with tools to enable more innovation, more quickly.

For more information, go to the Public Data Sets Page.

Available Public Data Sets

Public data sets are currently available in the following categories:

- **Biology—**Includes Human Genome Project, GenBank, and other content.
- **Chemistry—**Includes multiple versions of PubChem and other content.
- **Economics—**Includes census data, labor statistics, transportation statistics, and other content.
- **Encyclopedic—**Includes Wikipedia content from multiple sources and other content.

Finding Public Data Sets

Before you launch a public data set, you must locate it.

To find a public data set

1. Go to the Public Data Sets Page.

2. Locate a public data set and write down its snapshot ID for your operating platform (Windows or Linux/UNIX).

Launching an Instance

You'll attach a volume based on the public data set to an instance. Launch the instance as you typically launch an instance. For more information, see Launching Amazon EC2 Instances (p. 217).

Creating a Public Data Set Volume

To use a public data set, you create an Amazon EBS volume, specifying the snapshot ID of the public data set.

AWS Management Console

To create an Amazon EBS volume

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Click **Volumes** in the **Navigation** pane.
   The console displays a list of current volumes.
3. Click **Create Volume**.
   The **Create Volume** dialog box appears.
4. Configure the following settings and click **Create**.

   - The size of the volume, in GiB (optional)
   - The Availability Zone in which to launch the instance
   - The snapshot ID of the public data set

   Amazon EC2 begins creating the volume.

Command Line Tools

To create an Amazon EBS volume

1. Enter the following command.

```
PROMPT>  ec2-create-volume --snapshot public-data-set-snapshot-id --zone
availability-zone
```

   Amazon EBS returns information about the volume similar to the following example.

```
VOLUME vol-4d826724 85 us-east-1a creating 2008-02-14T00:00:00+0000
```

2. To check whether the volume is ready, use the following command.

```
PROMPT>  ec2-describe-volumes vol-4d826724
```

   Amazon EBS returns information about the volume similar to the following example.

```
VOLUME  vol-4d826724 85 us-east-1a  available 2008-07-29T08:49:25+0000
```

API

To create an Amazon EBS volume

- Construct the following Query request.

```
https://ec2.amazonaws.com/
?Action=CreateVolume
&AvailabilityZone=zone
&SnapshotId=public-data-set-snapshot-id
&AUTHPARAMS
```

The following is an example response.

```
<CreateVolumeResponse xmlns="http://ec2.amazonaws.com/doc/2012-07-20/">
  <volumeId>vol-4d826724</volumeId>
  <size>85</size>
  <status>creating</status>
  <createTime>2008-05-07T11:51:50.000Z</createTime>
  <availabilityZone>us-east-1a</availabilityZone>
  <snapshotId>snap-59d33330</snapshotId>
</CreateVolumeResponse>
```

Mounting the Public Data Set Volume

Mount the public data set volume as you typically mount an EBS volume. For more information, see
Making an Amazon EBS Volume Available for Use (p. 455).

Amazon EBS API and Command Overview

The following table summarizes the available Amazon EBS commands and corresponding API actions
for creating and using Amazon EBS volumes. For more information about the commands, go to the
Amazon Elastic Compute Cloud Command Line Reference. For more information about the API actions,
go to the Amazon Elastic Compute Cloud API Reference.

Command and API Action	Description
`ec2-create-volume` `CreateVolume`	Creates a new Amazon EBS volume using the specified size and type, or based on a previously created snapshot.
`ec2-delete-volume` `DeleteVolume`	Deletes the specified volume. The command does not delete any snapshots that were created from the volume.
`ec2-describe-volumes` `DescribeVolumes`	Describes your volumes, including size, volume type, source snapshot, Availability Zone, creation time, status (`available` or `in-use`). If the volume is `in-use`, an attachment line shows the volume ID, the instance ID to which the volume is attached, the device name exposed to the instance, its status (attaching, attached, detaching, detached), and when it attached.
`ec2-attach-volume` `AttachVolume`	Attaches the specified volume to a specified instance, exposing the volume using the specified device name. A volume can be attached to only a single instance at any time. The volume and instance must be in the same Availability Zone. The instance must be in the `running` or `stopped` state.

Command and API Action	Description
`ec2-detach-volume` `DetachVolume`	Detaches the specified volume from the instance it's attached to. This command does not delete the volume. The volume can be attached to another instance and will have the same data as when it was detached.
`ec2-create-snapshot` `CreateSnapshot`	Creates a snapshot of the volume you specify. After the snapshot is created, you can use it to create volumes that contain exactly the same data as the original volume.
`ec2-delete-snapshot` `DeleteSnapshot`	Deletes the specified snapshot. This command does not affect currently running Amazon EBS volumes, regardless of whether they were used to create the snapshot or were derived from the snapshot.
`ec2-describe-snapshots` `DescribeSnapshots`	Describes the specified snapshot. Describes all snapshots, including their source volume, snapshot initiation time, progress (percentage complete), and status (`pending`, `completed`, etc.).
`ec2-modify-snapshot-attribute` `ModifySnapshotAttribute`	Modifies permissions for a snapshot (i.e., who can create volumes from the snapshot). You can specify one or more AWS accounts, or specify `all` to make the snapshot public.
`ec2-describe-snapshot-attribute` `DescribeSnapshotAttribute`	Describes attributes for a snapshot.

Amazon EC2 Instance Store

Topics

Each Amazon EC2 instance, unless it's a micro instance, can access storage from disks that are physically attached to the host computer. This disk storage is referred to as *instance store*. Instance store provides temporary block-level storage for use with an Amazon EC2 instance. An instance store is dedicated to a particular instance; however, the disk subsystem is shared among instances on a host computer. The data in an instance store persists only during the lifetime of its associated Amazon EC2 instance.

Important

If an instance reboots (intentionally or unintentionally), data in the instance store persists. However, data on instance store volumes is lost under the following circumstances:

- Failure of an underlying drive
- Stopping an Amazon EBS-backed instance

- Terminating an instance

The size of an instance store ranges from 150 GiB to up to 3.3 TiB and varies by Amazon EC2 instance type. Larger Amazon EC2 instances have larger instance stores. For more information, see Instance Stores Available On Instance Types (p. 480).

An Amazon EC2 instance store consists of one or more instance store volumes. These volumes are usable only from a single Amazon EC2 instance during its lifetime; they can't be detached and then attached to another instance. Instance store volumes must be mounted before they can be used. By default, instances launched from an Amazon EBS-backed instance have no mounted instance store volumes. Instances launched from an instance store-backed AMI have a mounted instance store volume for the virtual machine's root device volume, and can have other mounted instances store volumes, depending on the instance type. For more information on instance store-backed AMIs and Amazon EBS-backed AMIs, see AMI Basics (p. 23).

When you launch an instance, whether it's launched from an Amazon EBS-backed AMI or an instance store-backed AMI, you can attach instance store volumes to the instance using block device mapping. For more information, see Block Device Mapping (p. 488).

If you need a permanent storage solution, or if you need to improve storage latency or throughput, we recommend using Amazon EBS volumes instead of instance store volumes. For more information, see Amazon Elastic Block Store (Amazon EBS) (p. 434).

Instance Stores Available On Instance Types

Amazon EC2 instances are divided into different instance types, which determine the size of the instance store available on the instance by default. When you launch an instance, you can specify an instance type or use the default instance type, which is an m1.small instance.

The instance type also determines the type of hardware for your instance store volumes. A high I/O instance (hi1.4xlarge) uses solid state drives (SSD) to deliver very high random I/O performance. This is a good option when you need storage with very low latency, but you don't need it to persist when the instance terminates, or you can take advantage of fault tolerant architectures. For more information see High I/O Instances (p. 97).

The following table shows the instance types along with the size of the instance store available with each instance type.

Type	Name	Instance Store Volumes
Micro	t1.micro	None (use Amazon EBS volumes)
Small	m1.small	1 x 150 GiB
Medium	m1.medium	1 x 400 GiB
Large	m1.large	2 x 420 GiB (840 GiB)
Extra Large	m1.xlarge	4 x 420 GiB (1680 GiB)
High-CPU Medium	c1.medium	1 x 340 GiB
High CPU Extra Large	c1.xlarge	4 x 420 GiB (1680 GiB)
High-Memory Extra Large	m2.xlarge	1 x 410 GiB
High-Memory Double Extra Large	m2.2xlarge	1 x 840 GiB

Type	Name	Instance Store Volumes
High-Memory Quadruple Extra Large	m2.4xlarge	2 x 840 GiB (1680 GiB)
High I/O	hi1.4xlarge	2 x 1 TiB SSD (2 TiB)
Cluster Compute Quadruple Extra Large	cc1.4xlarge	2 x 840 GiB (1680 GiB)
Cluster Compute Eight Extra Large	cc2.8xlarge	4 x 840 GiB (3360 GiB)
Cluster GPU Quadruple Extra Large	cg1.4xlarge	2 x 840 GiB (1680 GiB)

Instance Store Device Names

Within an instance store, instance store volumes are exposed as block devices. The virtual devices for instance store volumes are ephemeral[0-3]. Instance types that support 1 instance store volume have ephemeral0. Instance types that support 2 instance store volumes have ephemeral0 and ephemeral1. Instance types that support 4 instance store volumes have ephemeral0, ephemeral1, ephemeral2, and ephemeral3. Each instance store volume is pre-formatted with the ext3 file system. However, you can reformat volumes with the file system of your choice after you launch your instance. A Windows instance uses a built-in tool, EC2Config Service, to reformat the instance store volumes available on an instance with the NTFS file system.

Every instance store-backed AMI and instance has a mapping of the instance store volumes attached to the instance. Each entry in the mapping consists of a device name and the volume that it's mapped to. The instance store volumes are available to the instance, but you can't access them until they are mounted. A Windows instance uses the EC2Config Service to mount the instance store volumes for an instance. On Linux, the instance type determines which instance store volumes are mounted for you and which are available for you to mount yourself. The block device driver for the instance assigns the actual volume name when mounting the volume, and the name assigned can be different than the name that Amazon EC2 recommends.

The following table lists the devices names reserved for instance store volumes by instance type, and the default state of the storage device (formatted, mounted, available) on an instance store-backed instance. For example, an m1.small instance has ephemeral0 (ext3, 15 GiB) and swap (Linux swap, 896 MB).

Device Name	Linux/UNIX instance store-backed instance	Windows instance store-backed instance
/dev/sda1	Formatted and mounted as root (/).	Formatted and mounted as C:\.
/dev/sda2, xvdb	Formatted and mounted as /mnt or /media/ephemeral0 on m1.small and c1.medium instances.	Formatted and mounted on small instance types.
/dev/sda3	Formatted and mounted as /swap on m1.small and c1.medium instances.	Not available.
/dev/sdb, xvdb	Formatted and mounted as /mnt or /media/ephemeral0 on m1.medium, m1.large, m1.xlarge, c1.xlarge, cc1.4xlarge, cc2.8xlarge, m2.xlarge, m2.2xlarge, m2.4xlarge, and hi1.4xlarge.	Formatted and mounted on m1.medium, m1.large, m1.xlarge, c1.xlarge, m2.xlarge, and m2.2xlarge.

Device Name	Linux/UNIX instance store-backed instance	Windows instance store-backed instance
/dev/sdc, xvdc	Available on m1.large, m1.xlarge, cc2.8xlarge, cc1.4xlarge, c1.xlarge, and hi1.4xlarge.	Formatted and mounted on m1.large, m1.xlarge, c1.xlarge, and m2.4xlarge.
/dev/sdd, xvdd	Available on m1.xlarge, c1.xlarge, and cc2.8xlarge.	Formatted and mounted on m1.xlarge and c1.xlarge.
/dev/sde, xvde	Available on m1.xlarge and c1.xlarge.	Formatted and mounted on m1.xlarge and c1.xlarge.

An instance can have multiple instance store volumes mapped to a device. However, the number and size of these volumes must not exceed the instance store available for the instance type. For more information, see .

Amazon EC2 Instance Storage Usage Scenarios

Instance store volumes are ideal for temporary storage of information that changes frequently, such as buffers, caches, scratch data, and other temporary content, or for data that is replicated across a fleet of instances, such as a load-balanced pool of web servers.

Tip

Any data on an instance store volume is deleted if the instance stops, fails, or terminates. Therefore, do not rely on instance store volumes for valuable, long-term data. Instead, use a replication strategy across multiple instances to keep your data safe, store your persistent data in Amazon S3, or use Amazon EBS volumes for persistent data.

Tip

Data on instance store volumes is not included when you bundle an AMI. For example, if you have an instance store-backed Windows instance, the D: drive on that instance is typically an instance store volume, and it's not included in an AMI that you bundle from that instance.

Making Instance Stores Available on Your Instances

Instances that use Amazon EBS for the root device do not, by default, have instance store available at boot time. Also, you can't attach instance store volumes after you've launched an instance. Therefore, if you want your Amazon EBS-backed instance to use instance store volumes, you must specify them using a block device mapping when you create your AMI or launch your instance. Examples of block device mapping entries are: /dev/sdb=ephemeral0 and /dev/sdc=ephemeral1. For more information on block device mapping, see

The following procedure describes how to launch an Amazon EBS-backed *m1.large* Windows instance with instance store volumes.

Accessing Instance Stores on Amazon EBS-backed Windows Instances

1	Locate an Amazon EBS-backed Windows AMI.
2	Using the ec2-run-instances command, launch an *m1.large* instance and add block device mapping entries using the following options: -b "/dev/xvdb=ephemeral0" and -b "/dev/xvdc=ephemeral1".
3	Connect to the instance.

4	On the **Start** menu, choose **Computer**.
5	Devices listed: • Local Disk C:/ 9.98GiB • Local Disk D:/ 419GiB • Local Disk E:/ 419GiB
6	Double-click Local Disk C:/. You can see the list of installed applications. This is your root drive.
7	Double-click Local Disk D:/ and then double-click Local Disk E:/. These drives are empty. They are the instance stores that come with your *m1.large* instance, and they are available for you to use with your applications.

The following procedure describes how to launch an Amazon EBS-backed *m1.large* Linux instance with instance store volumes.

Accessing Instance Stores on Amazon EBS-backed Linux Instances

1	Locate an Amazon EBS-backed Linux/UNIX AMI.
2	Using the ec2-run-instances command, launch an *m1.large* instance and add block device mapping entries using the following options: -b "/dev/sdb=ephemeral0" and -b "/dev/sdc=ephemeral1".
3	Connect to the instance.
4	Verify the instance stores currently mounted on the disk.
5	Notice a 10GiB root partition mounted on the root and 420 GiB mounted on an ephemeral0 volume. Your *m1.large* instance comes with 2 420 GiB instance store volumes; the second volume is available but must be mounted before it can be used.
6	To format and mount the other 420 GiB instance store volume: a. Create a file system of your choice on the device /dev/sdc (requires root privileges). b. Create a directory on which to mount the device. c. Mount the device on the newly created directory.
7	Verify that the device has been mounted.
8	Optionally, list the files on the root device.

You can also map instance store volumes to block devices when you create an AMI. The instances launched from such an AMI have instance store volumes at boot time. For information on adding a block device mapping while creating an AMI, see the section called "Creating Your Own AMIs" (p. 35).

The following procedure describes how to access the instance store volumes from within an Amazon EC2 instance store-backed *m1.large* Windows instance.

Tasks for Accessing Instance Stores on Amazon EC2 instance store-backed Windows Instances

1	Locate an Amazon EC2 instance store-backed Windows AMI.

2	Launch an *m1.large* instance.
3	Connect to the instance.
4	On the **Start** menu, choose **Computer**.
5	Devices listed: • Local Disk C:/ 9.98GiB • Local Disk D:/ 419GiB • Local Disk E:/ 419GiB
6	Double-click Local Disk C:/. You see the list of all installed applications. This is your root drive.
7	Double-click Local Disk D:/ and then double-click Local Disk E:/. These are empty. They are the instance store volumes that come with your m1.large instance, and they are available to use with your applications just like any physical drive.

Depending on the instance type, some instance store volumes on Amazon EC2 instance store-backed Linux and UNIX instances are not mounted when the instance is launched. For example, on an m1.large Linux and UNIX instance, the device /dev/sdc, although formatted and available, must be mounted before it can be used.

The following procedure describes how to access the instance store from within Amazon EC2 instance store-backed *m1.large* Linux instance.

Accessing Instance Stores on Amazon EC2 instance store-backed Linux Instances

1	Locate an Amazon EC2 instance store-backed Linux/Unix AMI.
2	Launch an *m1.large* instance.
3	Connect to the instance.
4	Check out the file systems currently mounted on the disk.
5	Notice 10 GiB root partition mounted on the root and 420 GiB mounted on an ephemeral0 device. Your *m1.large* instance comes with 2 420 GiB instance store volumes; the second volume is available but must be mounted before it can be used.
6	To mount the other 420GiB: a. Create a directory on which to mount the device. b. Mount the device on the newly created directory.
7	Check to see whether the device has been mounted.
8	Optionally, list the files on the root device.

Suppressing Instance Stores at Launch Time

You can prevent a particular instance storage volume from attaching to the instance. You can do this for both Amazon EC2 instance store-backed instances and Amazon EBS-backed instances. For example,

specifying the mapping /dev/sdc=none when launching an instance prevents /dev/sdc from attaching to the instance. For more information on block device mapping, see Block Device Mapping (p. 488).

Using Amazon EC2 Instance Storage

This section describes how to use instance store volumes.

Adding A Default Instance Store

Amazon EBS-backed AMIs don't include instance store by default. However, you might want instances launched from your Amazon EBS-backed AMIs to include instance store. This section describes how to create an AMI that includes instance store.

Command Line Tools

Use the ec2-register command and specify a block device mapping that includes the instance store volumes for the image. For more information about block device mapping, see Block Device Mapping (p. 488).

> **Note**
>
> You cannot register an image where a secondary (non-root) snapshot has AWS Marketplace product codes.

To add instance storage by default

*   Use the ec2-register command with the desired block device mapping information.

    The following example registers an AMI with an 80GiB root device volume at /dev/sda1 created from the snap-12345678 snapshot. The root volume's `DeleteOnTermination` flag is set to `false`. The second block device mapping in the request maps /dev/sdc to ephemeral0.

    You can omit the size in the block device mapping if you want to create a volume that is the size of the snapshot. If you do specify a size, it must be equal to or larger than the size of the snapshot. You can also resize the partition or create a new partition later.

    > **Tip**
    >
    > If you're using the command line tools on a Windows computer, you must put quotation marks around any part of the command that includes an equal sign. For example: `... -b "/dev/sdc=ephemeral0" ....`

    ```
    PROMPT> ec2-register -n My_Image_Name -d My_image_description --root-device-
    name /dev/sda1  -b /dev/sda1=snap-12345678:80:false -b /dev/sdc=ephemeral0
    ```

    In response, you get the ID for your new AMI.

    ```
    IMAGE     ami-61a54008
    ```

Any instance you launch from this AMI includes the instance store volumes that you specified when you created the AMI. You can confirm that the instance store devices are available from within the instance itself using instance metadata. For more information, see Viewing Block Device Mappings (p. 491).

API

Use the RegisterImage action, specifying a block device mapping that includes the instance store volumes for the image.

Note

You cannot register an image where a secondary (non-root) snapshot has AWS Marketplace product codes.

To add instance storage by default

*   Issue the following Query request to register the image. The example registers an AMI with an 80 GiB root device volume at /dev/sda1 created from the snap-12345678 snapshot. The root volume's `deleteOnTermination` flag is set to `false`. The second block device mapping in the request maps /dev/sdc to ephemeral0.

    You can omit the size in the block device mapping if you want to create a volume that is the size of the snapshot. If you do specify a size, it must be equal to or larger than the size of the snapshot.

    ```
    https://ec2.amazonaws.com/
    ?Action=RegisterImage
    &Name=MyImage
    &KernelId=aki-f70657b2
    &RamdiskId=ari-ff0d5cba
    &RootDeviceName=/dev/sda1
    &BlockDeviceMapping.1.DeviceName=/dev/sda1
    &BlockDeviceMapping.1.Ebs.SnapshotId=snap-12345678
    &BlockDeviceMapping.1.Ebs.VolumeSize=80
    &BlockDeviceMapping.1.Ebs.DeleteOnTermination=false
    &BlockDeviceMapping.2.DeviceName=/dev/sdc
    &BlockDeviceMapping.2.VirtualName=ephemeral0
    &AUTHPARAMS
    ```

    For information about the *auth* parameters, go to Common Query Parameters in the *Amazon Elastic Compute Cloud API Reference*.

    The following is an example response.

    ```
    <RegisterImageResponse xmlns="http://ec2.amazonaws.com/doc/2012-07-20/">
       <imageId>ami-61a54008</imageId>
    </RegisterImageResponse>
    ```

Any instance you launch from this AMI includes the instance store volumes that you specified when you created the AMI. You can confirm the instance storage volumes are available from within the instance itself using instance metadata. For more information, see Viewing Block Device Mappings (p. 491)).

Disk Performance Optimization

Because of the way that Amazon EC2 virtualizes disks, the first write to any location on a standard instance store volume performs more slowly than subsequent writes. (The performance of the solid state drives (SSD) used by high I/O instances is not affected this way.) For most applications, amortizing this cost over the lifetime of the instance is acceptable. However, if you require high disk performance, we recommend that you initialize your drives by writing once to every drive location before production use. If you require further improvements in latency or throughput, we recommend using Amazon EBS.

To initialize the stores, use the following Linux/UNIX `dd` commands, depending on which store you want to initialize (/dev/sdb, etc.).

> **Note**
>
> Make sure to unmount the drive before performing this command.
>
> Initialization can take a long time (about 8 hours for an extra large instance).

To initialize the instance store volumes, use the following commands on the m1.large, m1.xlarge, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge 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
dd if=/dev/zero of=/dev/sde bs=1M
```

For information about the instance storage that is available for each instance type, see Instance Stores Available On Instance Types (p. 480).

To perform initialization on all instance store volumes at the same time, use the following command:

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

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
```

Amazon Simple Storage Service (Amazon S3)

Amazon Simple Storage Service (Amazon S3) is a repository for Internet data. Amazon S3 provides access to reliable, fast, and inexpensive data storage infrastructure. It is designed to make web-scale computing easy by enabling you to store and retrieve any amount of data, at any time, from within Amazon EC2 or anywhere on the web. Amazon S3 stores data objects redundantly on multiple devices across multiple facilities and allows concurrent read or write access to these data objects by many separate clients or application threads. You can use the redundant data stored in Amazon S3 to recover quickly and reliably from instance or application failures.

Objects are the fundamental entities stored in Amazon S3. Objects consist of object data and metadata. The metadata is a set of name-value pairs that describes the object. The data portion is opaque to Amazon S3.

Every object stored in Amazon S3 is contained in a bucket. Buckets organize the Amazon S3 namespace at the highest level and identify the account responsible for that storage. Amazon S3 buckets are similar to Internet domain names. Objects stored in the buckets have a unique key value and are retrieved using a HTTP URL address. For example, if an object with a key value /photos/mygarden.jpg is stored in the myawsbucket bucket, then it is addressable using the URL http://myawsbucket.s3.amazonaws.com/photos/mygarden.jpg.

For more information on Amazon S3 go to the Amazon Simple Storage Service Developer Guide.

Amazon S3 Usage Scenario

The combination of Amazon S3 and Amazon EC2 provides resizable computing capacity with highly scalable and fast data storage infrastructure.

Amazon EC2 uses Amazon S3 for storing Amazon Machine Images (AMIs). You use AMIs for launching EC2 instances. In case of instance failure, you can use the stored AMI to immediately launch another instance, thereby allowing for fast recovery and business continuity.

Amazon EC2 also uses Amazon S3 to store snapshots (backup copies) of the data volumes. You can use snapshots for recovering data quickly and reliably in case of application or system failures. You can also use snapshots as a baseline to create multiple new data volumes, expand the size of an existing data volume, or move data volumes across multiple Availability Zones, thereby making your data usage highly scalable. For more information on using data volumes and snapshots, see Amazon Elastic Block Store (p. 434).

If you already have your data stored in Amazon S3 and want to move it to Amazon EBS volume, you can do so by mounting an empty Amazon EBS volume to an Amazon EC2 instance and then using a data transfer application (e.g., FTP/SFTP, SCP) on your running instance.

Amazon S3 stores AMIs and snapshots redundantly on multiple devices across multiple facilities, thus providing an Amazon EC2 instance with highly durable storage infrastructure.

Block Device Mapping

Topics

- Block Device Mapping Concepts (p. 488)
- AMI Block Device Mapping (p. 491)
- Instance Block Device Mapping (p. 495)

Each Amazon EC2 instance that you launch has an associated root device volume, either an Amazon Elastic Block Store (EBS) volume or an instance store volume. You can use block device mapping to specify additional EBS volumes or instance store volumes to attach to an instance when it's launched. You can also attach additional EBS volumes to a running instance; see Attaching a Volume to an Instance (p. 450). However, the only way to attach instance store volumes to an instance is to use block device mapping to attach them as the instance is launched.

For more information about root device volumes, see Using Amazon EC2 Root Device Storage (p. 122).

Block Device Mapping Concepts

A *block device* is a storage device that moves data in sequences of bytes or bits (blocks). These devices support random access and generally use buffered I/O. Examples include hard disks, CD-ROM drives, and flash drives. A block device can be physically attached to a computer or accessed remotely as if it were physically attached to the computer. Amazon EC2 supports two types of block devices:

- instance store volumes (virtual devices whose underlying hardware is physically attached to the host computer for the instance)
- Amazon EBS volumes (remote storage devices)

A *block device mapping* defines the block devices to be attached to an Amazon EC2 instance. You can specify a block device mapping as part of creating an AMI so that the mapping is used by all instances

launched from the AMI. Alternatively, you can specify a block device mapping when you launch an instance, so this mapping overrides the one specified in the AMI from which you launched the instance.

Specifying a Block Device Mapping

Use a block device mapping to attach instance store volumes and EBS volumes to an Amazon EC2 instance.

When you create a block device mapping, you specify this information for each block device that you need to attach to the instance:

- [Linux/UNIX] The device name within Amazon EC2, as shown in this table. The block device driver for the instance assigns the actual volume name when mounting the volume, and the name assigned can be different than the name that Amazon EC2 recommends.

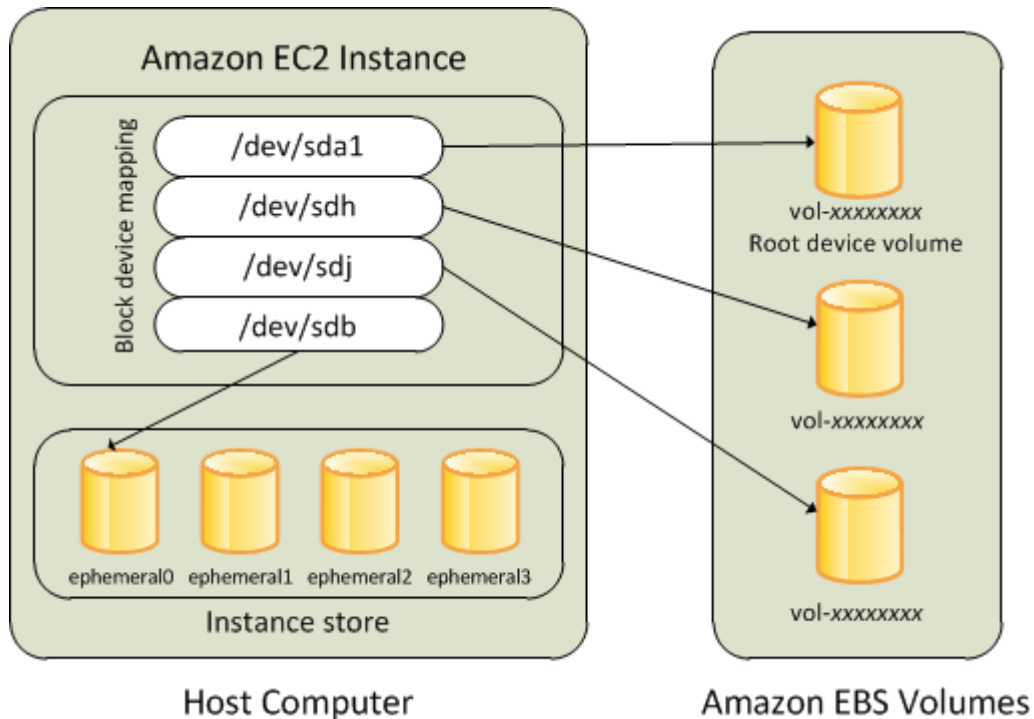|	
Reserved for the root device	/dev/sda1
Recommended for instance store volumes	/dev/sd[b-e]
Recommended for EBS volumes	/dev/sd[f-p] /dev/sd[f-p][1-6]
Possible for EBS volumes	/dev/sd[a-z] /dev/sd[a-z][1-15] /dev/hd[a-z] /dev/hd[a-z][1-15]

- [Windows] The device name within Amazon EC2, as shown in this table. The block device driver for the instance assigns the actual volume name when mounting the volume, and the name assigned can be different than the name that Amazon EC2 recommends.

Reserved for the root device	/dev/sda1
Recommended for instance store volumes	xvd[a-e]
Recommended for EBS volumes	xvd[f-p]
Possible for EBS volumes	xvd[a-p] /dev/sda[1-2] /dev/sd[b-e]

- [Instance store volumes only] The virtual device: ephemeral[0-3].
- [EBS volumes only] The ID of the snapshot to use to create the block device (snap-*xxxxxxxx*). This value is optional as long as you specify a volume size.
- [EBS volumes only] The size of the volume, in GiB. The specified size must be greater than or equal to the size of the specified snapshot.
- [EBS volumes only] Whether to delete the volume on instance termination (`true` or `false`). The default value is `true`.
- [EBS volumes only] The volume type (`standard` or `io1`). The default value is `standard`.
- [EBS volumes only] The number of input/output operations per second (IOPS) that the volume supports. (Not used with `standard` volumes.)

Example Block Device Mapping

This figure shows an example block device mapping for an Amazon EBS-backed instance. It maps /dev/sdb to ephemeral0 and maps two EBS volumes, one to /dev/sdh and the other to /dev/sdj. It also shows the EBS volume that is the root device volume, /dev/sda1.

Note that this example block device mapping is used in the example commands and APIs in this topic. You can find example commands and APIs that create block device mappings here:

- Specifying a Block Device Mapping for an AMI (p. 491)
- Updating the Block Device Mapping when Launching an Instance (p. 495)

How Devices are Made Available in the Operating System

Device names like /dev/sdh and xvdh are used by Amazon EC2 to describe block devices. The block device mapping is used by Amazon EC2 to specify the block devices to attach to an Amazon EC2 instance. After a block device is attached to an instance, it must be mounted by the operating system before you can access the storage device. When a block device is detached from an instance, it is unmounted by the operating system and you can no longer access the storage device.

With a Linux instance, the device names specified in the block device mapping are mapped to their corresponding block devices when the instance first boots. The instance type determines which instance store volumes are formatted and mounted by default. You can mount additional instance store volumes at launch, as long as you don't exceed the number of instance store volumes available for your instance type. For more information, see Amazon EC2 Instance Store (p. 479). The block device driver for the instance determines which devices are used when the volumes are formatted and mounted. For more information, see Attaching a Volume to an Instance (p. 450).

With a Windows instance, the device names specified in the block device mapping are mapped to their corresponding block devices when the instance first boots, and then the Ec2Config service initializes and mounts the drives. The root device volume is mounted as C:\. The instance store volumes are mounted

as D:\, E:\, and so on. When an EBS volume is mounted, it can be mounted using any available drive letter. However, you can configure how the Ec2Config Service assigns drive letters to EBS volumes; for more information, see Using EC2Config Service.

Viewing Block Device Mappings

You can view information about each block device in a block device mapping. For details, see:

AMI Block Device Mapping

Each AMI has a block device mapping that specifies the block devices to attach to an instance when it is launched from the AMI. An AMI that Amazon provides includes a root device only. To add additional block devices to an AMI, you must create your own AMI.

Specifying a Block Device Mapping for an AMI

For an EBS-backed AMI, you can add EBS volumes and instance store volumes using a block device mapping. For an instance store-backed AMI, you can add only instance store volumes using a block device mapping.

AWS Management Console

If you've already attached volumes to a running instance, you can create an AMI from the instance, which gives the AMI a block device mapping that attaches those same volumes. Follow the steps in Creating Amazon EBS-Backed AMIs Using the Console (p. 33).

Command Line Tools

Use **ec2-register -b** "*devicename*=*blockdevice*" to create an AMI with a block device mapping.

devicename
    The device name within Amazon EC2

blockdevice
    To omit a mapping for the device from the AMI, specify `none`.

    To add an instance store device, specify `ephemeral[0-3]`.

    [EBS-backed instance only] To add an EBS volume, specify *snapshot-id*:*size*:[`true`|`false`]. To add an empty EBS volume, omit the snapshot ID. To indicate whether the EBS volume should be deleted on termination, specify true or false; the default value is true.

For example, run this command at a command prompt to register an Amazon EBS-backed Linux AMI with an instance store volume, an EBS volume based on a snapshot, and an empty 100 GiB EBS volume. (Be sure to set the `EC2_PRIVATE_KEY` and `EC2_CERT` environment variables first.)

```
ec2-register -n ImageName --root-device-name /dev/sda1 -s snap-e1eb279f
-b "/dev/sdb=ephemeral0" -b "/dev/sdh=snap-d5eb27ab" -b "/dev/sdj=:100"
```

This command maps /dev/sdc to ephemeral0, /dev/sdh to an EBS volume based on a snapshot, and /dev/sdj to an empty EBS volume that is 100 GiB in size. The output is the ID for your new AMI.

```
IMAGE ami-72aa081b
```

For more information, see ec2-register.

To verify that the AMI was created successfully, look for it in the Amazon EC2 Console (click **AMIs** in the **Navigation** pane, then click **Owned By Me** in the **Viewing** menu) or the output of the following command:

```
ec2-describe-images -o self
```

Alternatively, if you've already attached volumes to an instance, you can use one of these methods as appropriate.

- Amazon EBS-backed AMI
  - ec2-create-image
- Instance store-backed AMI
  - Creating an Instance Store-Backed AMI From an Existing AMI (p. 41)
  - Creating an Instance Store-Backed Windows AMI

API

Use `RegisterImage` to create an AMI with a block device mapping.

For example, use this request to register a Linux AMI with its root device, an instance store volume, an EBS volume based on a snapshot, and an empty 100 GiB EBS volume.

```
https://ec2.amazonaws.com/?Action=RegisterImage
&Name=ImageName
&RootDeviceName=/dev/sda1
&BlockDeviceMapping.1.DeviceName=/dev/sdb
&BlockDeviceMapping.1.VirtualName=ephemeral0
&BlockDeviceMapping.2.DeviceName=/dev/sdh
&BlockDeviceMapping.2.Ebs.SnapshotId=snap-d5eb27ab
&BlockDeviceMapping.3.DeviceName=/dev/sdj
&BlockDeviceMapping.3.Ebs.VolumeSize=100
&AUTHPARAMS
```

Note that AUTHPARAMS represents common required parameters. For more information, see Common Query Parameters or Making Query Requests (p. 533).

This request maps /dev/sdc to ephemeral0, /dev/sdh to an EBS volume based on a snapshot, and /dev/sdj to an empty EBS volume that is 100 GiB in size. The response includes the ID for your new AMI.

```
...
  <imageId>ami-72aa081b</imageId>
...
```

For more information, see RegisterImage.

To verify that the AMI was created successfully, look for it in the Amazon EC2 Console (click **AMIs** in the **Navigation** pane, then click **Owned By Me** in the **Viewing** menu) or the output of the following command:

```
ec2-describe-images -o self
```

Alternatively, if you've already attached volumes to an Amazon EBS-backed instance, you can use CreateImage to create an AMI from this instance with a block device mapping that attaches those same volumes.

Viewing the EBS Volumes in an AMI Block Device Mapping

You can easily enumerate the EBS volumes in the block device mapping for an AMI.

AWS Management Console

Use the AWS Management Console as follows to enumerate the EBS volumes in the block device mapping for an AMI:

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/
2. In the **Navigation** pane, click **AMIs**.
3. In the **Viewing** menu, click **EBS Images** to filter the list. This displays a list of Amazon EBS-backed AMIs.
4. Locate and click the desired AMI and look at the details displayed in the **EC2 Amazon Machine Image** pane. At a minimum, the following information is available for the root device:

   - **Root Device Type** (ebs)
   - **Root Device** (for example, /dev/sda1)
   - **Block Devices** (for example, /dev/sda1=snap-e1eb279f:8:true)

   If the AMI was created with additional EBS volumes using a block device mapping, the **Block Devices** box displays the mapping for those additional EBS volumes as well. (Recall that this dialog box doesn't display instance store volumes.)

Command Line Tools

Use **ec2-describe-images** *ami_id* to enumerate the EBS volumes in the block device mapping for an AMI.

For example, run this command at a command prompt to get information about an AMI, including its block device mapping. (Be sure to set the EC2_PRIVATE_KEY and EC2_CERT environment variables first.)

```
ec2-describe-images ami-72aa081b
```

The output includes the block device mapping for your AMI. The following information is available for EBS volumes:

- The device name within Amazon EC2
- The ID of the snapshot used when creating the block device (optional)
- The size of the volume, in GiB
- The volume type.

Here's example output for a Linux AMI.

```
BLOCKDEVICEMAPPING    /dev/sda1    snap-e1eb279f    8      standard
BLOCKDEVICEMAPPING    /dev/sdh     snap-d5eb27ab    200    standard
BLOCKDEVICEMAPPING    /dev/sdj                      100    standard
```

For more information, see ec2-describe-images.

API

Use `DescribeImages` to enumerate the EBS volumes in the block device mapping for an AMI.

For example, use this request to get information about an AMI, including its block device mapping.

```
https://ec2.amazonaws.com/?Action=DescribeImages
&ImageId.1=ami-72aa081b
&AUTHPARAMS
```

Note that AUTHPARAMS represents common required parameters. For more information, see Common Query Parameters or Making Query Requests (p. 533).

The response includes the block device mapping for your AMI. The following information is available for EBS volumes:

- The device name within Amazon EC2
- The ID of the snapshot used when creating the block device (optional)
- The size of the volume, in GiB
- Whether to delete the volume on instance termination (default is true)
- The volume type.

Here's an example response for a Linux AMI.

```
...
  <blockDeviceMapping>
    <item>
      <deviceName>/dev/sda1</deviceName>
      <ebs>
        <snapshotId>snap-e1eb279f</snapshotId>
        <volumeSize>8</volumeSize>
```

```
            <deleteOnTermination>true</deleteOnTermination>
            <volumeType>standard</volumeType>
          </ebs>
        </item>
        <item>
          <deviceName>/dev/sdh</deviceName>
          <ebs>
            <snapshotId>snap-d5eb27ab</snapshotId>
            <volumeSize>200</volumeSize>
            <deleteOnTermination>true</deleteOnTermination>
            <volumeType>standard</volumeType>
          </ebs>
        </item>
        <item>
          <deviceName>/dev/sdj</deviceName>
          <ebs>
            <volumeSize>100</volumeSize>
            <deleteOnTermination>true</deleteOnTermination>
            <volumeType>standard</volumeType>
          </ebs>
        </item>
      </blockDeviceMapping>
...
```

For more information, see DescribeImages.

Instance Block Device Mapping

By default, an instance that you launch includes any storage devices specified in the block device mapping of the AMI from which you launched the instance. You can specify changes to the block device mapping for an instance when you launch it, and these updates overwrite or merge with the block device mapping of the AMI. However, you can't modify the block device mapping entry for the root device volume.

Updating the Block Device Mapping when Launching an Instance

You can add EBS volumes and instance store volumes to an instance when you launch it. Note that updating the block device mapping for an instance doesn't make a permanent change to the block device mapping of the AMI from which it was launched.

AWS Management Console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/
2. From the Amazon EC2 console dashboard, click **Launch Instance**.
3. On the **Create a New Instance** page, click **Quick Launch Wizard**. Fill in the name of the instance and the name of the key pair as usual, select an AMI, and click **Continue** to view your settings.
4. Click **Edit details**.
5. Click **Storage Device Configuration** to edit the block device mapping. You can modify the root volume, EBS volumes, and instance store volumes as follows:

   - To change the size of the root volume, click **Root Volume**, fill in the **Volume Size** field, and click **Save**.
   - To suppress an EBS volume specified by the block device mapping of the AMI used to launch the instance, click **EBS Volumes**, go to the volume, and click **Delete**.

- To add an EBS volume, click **EBS Volumes**, fill in the fields (**Device**, **Snapshot**, and so on), and click **Add**.
- To suppress an instance store volume specified by the block device mapping of the AMI used to launch the instance, click **Instance Store Volumes**, go to the volume, and click **Delete**.
- To add an instance store volume, click **Instance Store Volumes**, select the **Instance Store** and **Device**, and click **Add**.

6. When you've finished modifying the block device mapping, click **Save details**.

Command Line Tools

Use **ec2-run-instances -b** "*devicename*=*blockdevice*" to define an entry for the block device mapping for an instance.

devicename
 The device name within Amazon EC2

blockdevice
 To omit a mapping for the device from the AMI, specify `none`.

 To add an instance store device, specify `ephemeral[0-3]`.

 [EBS-backed instance only] To add an EBS volume, specify *snapshot-id*:*size*:[`true|false`]. To add an empty EBS volume, omit the snapshot ID. To indicate whether the EBS volume should be deleted on termination, specify true or false; the default value is true.

For example, suppose that an EBS-backed Linux AMI specifies the following block device mapping:

- /dev/sdb=ephemeral0
- /dev/sdh=snap-92d333fb
- /dev/sdj=:100

To prevent /dev/sdj from attaching to an instance launched from this AMI, use this option.

```
-b "/dev/sdj=none"
```

To increase the size of /dev/sdh to 300 GiB, use this option.

```
-b "/dev/sdh=:300"
```

Notice that we didn't need to specify the snapshot ID for /dev/sdh, because specifying the device name is enough to identify the volume.

To attach an additional instance store volume, /dev/sdc, use this option. If the instance type doesn't support more than one instance store volume, this option has no effect.

```
-b "/dev/sdc=ephemeral1"
```

For more information, see ec2-run-instances.

API

Use `RunInstances` to define an entry for the block mapping for an instance.

For example, suppose that an EBS-backed Linux AMI specifies the following block device mapping:

- /dev/sdb=ephemeral0
- /dev/sdh=snap-92d333fb
- /dev/sdj=:100

Use this request to do the following:

- prevent /dev/sdj from attaching to an instance launched from this AMI
- increase the size of /dev/sdh to 300 GiB
- attach an additional instance store volume, /dev/sdc (if the instance type supports it)

```
http://ec2.amazonaws.com/?Action=RunInstances
&ImageId.1=ami-72aa081b
...
&BlockDeviceMapping.1.DeviceName=/dev/sdj
&BlockDeviceMapping.1.Ebs.NoDevice=true
&BlockDeviceMapping.2.DeviceName=/dev/sdh
&BlockDeviceMapping.2.Ebs.VolumeSize=300
&BlockDeviceMapping.3.DeviceName=/dev/sdc
&BlockDeviceMapping.3.VirtualName=ephemeral1
&AUTHPARAMS
```

Note that AUTHPARAMS represents common required parameters. For more information, see Common Query Parameters or Making Query Requests (p. 533).

Notice that we didn't need to specify the snapshot ID for /dev/sdh, because specifying the device name is enough to identify the volume.

For more information, see RunInstances.

Viewing the EBS Volumes in an Instance Block Device Mapping

You can easily enumerate the EBS volumes mapped to an instance.

Note

For instances launched before the release of the 2009-10-31 API, AWS can't display the block device mapping. You must detach and reattach the volumes so that AWS can display the block device mapping.

AWS Management Console

Use the AWS Management Console to enumerate the EBS volumes in the block device mapping for an instance.

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/
2. In the **Navigation** pane, click **Instances**.
3. In the **Viewing** menu, click **EBS Root Device** to filter the list. This displays a list of Amazon EBS-backed instances.
4. Locate and click the desired instance and look at the details displayed in the **EC2 Instances** pane. At a minimum, the following information is available for the root device:

- **Root Device Type** (ebs)
- **Root Device** (for example, sda1)
- **Block Devices** (for example, sda1, sdh, and sdj)

If the instance was launched with additional EBS volumes using a block device mapping, the **Block Devices** box displays those additional EBS volumes as well as the root device. (Recall that this dialog box doesn't display instance store volumes.)

5. To display additional information about a block device, click its entry under Block Devices. This displays the following information for the block device:

- EBS ID (vol-*xxxxxxxx*)
- **Root Device Type** (ebs)
- **Attachment time** (*yyyy-mm*T*hh*:*mm*:*ss.ssTZD*)
- **Block device status** (attaching, attached, detaching, detached)
- **Delete on termination** (Yes, No)
- **Snapshot ID** (snap-*xxxxxxxx*)

Command Line Tools

Use **ec2-describe-instances** *instance_id* to enumerate the EBS volumes in the block device mapping for an instance.

For example, run this command at a command prompt to get information about an instance, including its block devices. (Be sure to set the EC2_PRIVATE_KEY and EC2_CERT environment variables first.)

```
ec2-describe-instances i-xxxxxxxx
```

The output includes the block devices for your instance. The following information is available for EBS volumes:

- The device name within Amazon EC2
- The volume ID
- The time when the attachment was initiated
- Whether to delete the volume on instance termination
- The volume type (`standard` or `io1`)
- The number of input/output operations per second (IOPS) that the volume supports. (Not used with `standard` volumes.)

The output for a Linux instance looks something like this.

```
BLOCKDEVICE     /dev/sda1     vol-xxxxxxxx     yyyy-mmThh:mm:ss.sTZD     true
BLOCKDEVICE     /dev/sdh      vol-xxxxxxxx     yyyy-mmThh:mm:ss.sTZD     true
BLOCKDEVICE     /dev/sdj      vol-xxxxxxxx     yyyy-mmThh:mm:ss.sTZD     true
```

To view this information plus the status (attaching, attached, detaching, detached), run the **ec2-describe-instances** command with the **–v** option to see the full SOAP response.

For more info, see ec2-describe-instances.

API

Use `DescribeInstances` to enumerate the EBS volumes in the block device mapping for an instance.

For example, use this request to get information about an instance, including its block devices.

```
https://ec2.amazonaws.com/?Action=DescribeInstances
&InstanceId.1=i-xxxxxxxx
&AUTHPARAMS
```

Note that AUTHPARAMS represents common required parameters. For more information, see Common Query Parameters or Making Query Requests (p. 533).

The response includes the block devices for your instance. The following information is available for EBS volumes:

- The device name within Amazon EC2
- The volume ID
- The status (attaching, attached, detaching, detached)
- The time when the attachment was initiated
- Whether to delete the volume on instance termination
- The volume type (`standard` or `io1`)
- The number of input/output operations per second (IOPS) that the volume supports. (Not used with `standard` volumes.)

Here's example output for a Linux instance.

```
...
  <blockDeviceMapping>
    <item>
      <deviceName>/dev/sda1</deviceName>
      <ebs>
        <volumeId>vol-xxxxxxxx</volumeId>
        <status>attached</status>
        <attachTime>yyyy-mm-ddThh:mm:ss.sTZD</attachTime>
        <deleteOnTermination>true</deleteOnTermination>
        <volumeType>standard</volumeType>
      </ebs>
    </item>
    <item>
      <deviceName>/dev/sdh</deviceName>
      <ebs>
        <volumeId>vol-xxxxxxxx</volumeId>
        <status>attached</status>
        <attachTime>yyyy-mm-ddThh:mm:ss.sTZD</attachTime>
        <deleteOnTermination>true</deleteOnTermination>
        <volumeType>standard</volumeType>
      </ebs>
    </item>
    <item>
      <deviceName>/dev/sdj</deviceName>
      <ebs>
        <volumeId>vol-xxxxxxxx</volumeId>
```

```
        <status>attached</status>
        <attachTime>yyyy-mm-ddThh:mm:ss.sTZD</attachTime>
        <deleteOnTermination>true</deleteOnTermination>
        <volumeType>standard</volumeType>
      </ebs>
    </item>
  </blockDeviceMapping>
...
```

For more info, see DescribeInstances.

Viewing the Instance Block Device Mapping for Instance Store Volumes

When you view the block device mapping for your instance, you can see only the EBS volumes, not the instance store volumes. You can use instance metadata to query the complete block device mapping. The base URI for all requests for instance metadata is `http://169.254.169.254/latest/`.

First, connect to your running instance. If the instance is running Linux, the GET command is already available. If the instance is running Windows, install wget on the instance, and replace GET with wget in the examples below.

Use this query on a running instance to get its block device mapping.

```
GET http://169.254.169.254/latest/meta-data/block-device-mapping/
```

The response includes the names of the block devices for the instance. For example, the output for an instance store-backed m1.small instance looks like this.

```
ami
ephemeral0
root
swap
```

The ami device is the root device as seen by the instance. The instance store volumes are named ephemeral[0-3]. The swap device is for the page file. If you've also mapped EBS volumes, they appear as ebs1, ebs2, and so on.

To get details about an individual block device in the block device mapping, append its name to the previous query, as shown here.

```
GET http://169.254.169.254/latest/meta-data/block-device-mapping/ephemeral0
```

The response includes the device name, as shown here.

```
/dev/sdb
```

For more info, see Instance Metadata (p. 302).

Resources and Tags

Topics

This section describes key features that enable you to manage Amazon EC2 resources, such as AMIs, instances, Amazon EBS volumes, and snapshots.

Resource Locations

The following table describes which Amazon EC2 resources are global, Regional, or Availability Zone-based.

Resource	Type	Description
AWS Account	Global	You use the same AWS account in all Regions.
DevPay Product Codes	Global	You use the same DevPay product codes throughout all Regions.
Amazon EC2 System Identifiers	Regional	Includes the AMI ID, Instance ID, EBS Volume ID, EBS Snapshot ID, and so on.
Instances	Availability Zone	Instances are tied to Availability Zones. However, the instance ID is tied to the Region.
AMIs	Regional	AMIs are tied to the Region where its files are located within Amazon S3.
Security Groups	Regional	Security groups are not copied across Regions. Instances within the Region cannot communicate with instances outside the Region using group-based firewall rules. Traffic from instances in another Region is seen as WAN bandwidth.

Resource	Type	Description
SSH Key Pairs	Regional or Global	The SSH key pairs that you create with `ec2-add-keypair`, `CreateKeyPair`, or in the AWS Management Console work only in the Region where you create them. However, you can optionally create an RSA key pair with a third-party tool and upload the public key to AWS. That key pair works in all Regions. For more information, go to ec2-import-keypair in the *Amazon Elastic Compute Cloud Command Line Reference* or ImportKeyPair in the *Amazon Elastic Compute Cloud API Reference*.
User-Supplied Identifiers	Regional	Includes security group names, SSH key pair names, and so on. Although you can create the same names in multiple Regions, they have no relationship to each other.
Elastic IP Addresses	Regional	Elastic IP addresses are tied to a Region and cannot be mapped across Regions.
EBS Volumes	Availability Zone	An Amazon EBS volume must be located within the same Availability Zone as the instance to which it attaches.
EBS Snapshots	Regional	Snapshots are tied to Regions and can only be used for volumes within the same Region.

Listing and Filtering Your Resources

Topics

As part of using Amazon EC2, you might use different *resources*. These includes images, instances, Amazon EBS volumes, snapshots, etc. You can list each type of resource with the corresponding *describe* action (e.g., `ec2-describe-images`, `ec2-describe-instances`). The resulting list can be long, so you might want to filter the results to include only resources that match certain criteria. For example, you could list only the public 64-bit Windows AMIs that use an Amazon EBS volume as the root device. You could list only instances that have an attached Amazon EBS volume that is set to delete whenever the instance terminates. You could list only Amazon EBS snapshots that have been tagged with a certain value.

The EC2 command line interface (also called the API tools) and API let you specify filters when listing resources. The specific filters you can use are listed in the topic covering the particular describe call. The examples that follow show some of the filters.

You can specify multiple values for a filter. For example, you can list all the instances whose type is either m1.small or m1.large. The resource must match at least one of the values to be included in the resulting resource list. You can also specify multiple filters. For example, you can list all the instances whose type is either m1.small or m1.large, and that have an attached EBS volume that is set to delete when the instance terminates. The instance must match all your filters to be included in the results.

You can also use wildcards with the filter values. An asterisk (*) matches zero or more characters, and a question mark (?) matches exactly one character. For example, you can use `*database*` as a filter value to get all EBS snapshots that include `database` in the description. If you were to specify `database` as the filter value, then only snapshots whose description equals `database` would be returned. Filter values are case sensitive. We support only exact string matching, or substring matching (with wildcards).

> **Note**
>
> Your search can include the literal values of the wildcard characters; you just need to escape them with a backslash before the character. For example, a value of `\*amazon\?\\` searches for the literal string `*amazon?\`.

Listing Resources

Each resource type has a corresponding *describe* action that you use to list your resources of that type. For example, you use `ec2-describe-images` or `ec2-describe-instances` to list your images or instances, respectively.

Command Line Tools

The following example lists your instances.

```
PROMPT> ec2-describe-instances
```

The response contains information about all your instances. Following is a sample response:

```
RESERVATION        r-c482cbaf        111122223333      default
INSTANCE           i-8f59c3e5        ami-c5e40dac      ec2-184-73-22-106.compute-
```

```
1.amazonaws.com        ip-10-194-250-85.ec2.internal   running   GSG_Keypair
0            m1.small      2010-08-17T14:26:03+0000        us-east-1d
    windows   monitoring-disabled    184.73.22.106   10.194.250.85
  ebs       hvm
BLOCKDEVICE     /dev/sda1      vol-6214560b    2010-08-17T14:26:07.000Z
RESERVATION     r-f0ddf49b     111122223333    default
INSTANCE        i-59d69933     ami-b232d0db    ec2-75-101-210-167.compute-
1.amazonaws.com     domU-12-31-39-0F-CD-55.compute-1.internal      running
    GSG_Keypair    0            m1.small        2010-09-03T16:09:09+0000
      us-east-1b  aki-94c527fd      ari-96c527ff         monitoring-dis
abled   75.101.210.167   10.193.206.163                ebs    spot    sir-
e1aa2e03            paravirtual
BLOCKDEVICE     /dev/sda1      vol-28421e41    2010-09-03T16:09:16.000Z
```

API

The following Query example lists your instances.

```
https://ec2.amazonaws.com/?Action=DescribeInstances
&AuthParams
```

Here is a sample response to the list request:

```
<DescribeInstancesResponse xmlns="http://ec2.amazonaws.com/doc/2012-07-20/">
  <requestId>98e3c9a4-848c-4d6d-8e8a-b1bdEXAMPLE</requestId>
  <reservationSet>
    <item>
      <reservationId>r-b27e30d9</reservationId>
      <ownerId>111122223333</ownerId>
      <groupSet>
        <item>
          <groupId>sg-2eac845a</groupId>
          <groupName>default</groupName>
        </item>
      </groupSet>
      <instancesSet>
        <item>
          <instanceId>i-c5cd56af</instanceId>
          <imageId>ami-1a2b3c4d</imageId>
          <instanceState>
            <code>16</code>
            <name>running</name>
          </instanceState>
         <privateDnsName>domU-12-31-39-10-56-34.compute-1.internal</privateDns
Name>
          <dnsName>ec2-174-129-165-232.compute-1.amazonaws.com</dnsName>
          <reason/>
          <keyName>GSG_Keypair</keyName>
          <amiLaunchIndex>0</amiLaunchIndex>
          <productCodes/>
          <instanceType>m1.small</instanceType>
          <launchTime>2010-08-17T01:15:18.000Z</launchTime>
          <placement>
            <availabilityZone>us-east-1b</availabilityZone>
            <groupName/>
```

```
            </placement>
            <kernelId>aki-94c527fd</kernelId>
            <ramdiskId>ari-96c527ff</ramdiskId>
            <monitoring>
               <state>disabled</state>
            </monitoring>
            <privateIpAddress>10.198.85.190</privateIpAddress>
            <ipAddress>174.129.165.232</ipAddress>
            <sourceDestCheck>true</sourceDestCheck>
            <groupSet>
               <item>
                  <groupId>sg-2eac845a</groupId>
                  <groupName>default</groupName>
               </item>
            </groupSet>
            <architecture>i386</architecture>
            <rootDeviceType>ebs</rootDeviceType>
            <rootDeviceName>/dev/sda1</rootDeviceName>
            <blockDeviceMapping>
              <item>
                <deviceName>/dev/sda1</deviceName>
                <ebs>
                  <volumeId>vol-a082c1c9</volumeId>
                  <status>attached</status>
                  <attachTime>2010-08-17T01:15:21.000Z</attachTime>
                  <deleteOnTermination>false</deleteOnTermination>
                </ebs>
              </item>
            </blockDeviceMapping>
            <instanceLifecycle>spot</instanceLifecycle>
            <spotInstanceRequestId>sir-7a688402</spotInstanceRequestId>
            <virtualizationType>paravirtual</virtualizationType>
            <clientToken/>
            <tagSet/>
            <hypervisor>xen</hypervisor>
       </item>
     </instancesSet>
     <requesterId>854251627541</requesterId>
   </item>
   <item>
     <reservationId>r-b67e30dd</reservationId>
     <ownerId>111122223333</ownerId>
     <groupSet>
       <item>
          <groupId>sg-2eac845a</groupId>
         <groupId>default</groupId>
       </item>
     </groupSet>
     <instancesSet>
       <item>
         <instanceId>i-d9cd56b3</instanceId>
         <imageId>ami-1a2b3c4d</imageId>
         <instanceState>
           <code>16</code>
           <name>running</name>
         </instanceState>
        <privateDnsName>domU-12-31-39-10-54-E5.compute-1.internal</privateDns
Name>
```

```
              <dnsName>ec2-184-73-58-78.compute-1.amazonaws.com</dnsName>
              <reason/>
              <keyName>GSG_Keypair</keyName>
              <amiLaunchIndex>0</amiLaunchIndex>
              <productCodes/>
              <instanceType>m1.large</instanceType>
              <launchTime>2010-08-17T01:15:19.000Z</launchTime>
              <placement>
                 <availabilityZone>us-east-1b</availabilityZone>
                 <groupName/>
              </placement>
              <kernelId>aki-94c527fd</kernelId>
              <ramdiskId>ari-96c527ff</ramdiskId>
              <monitoring>
                 <state>disabled</state>
              </monitoring>
              <privateIpAddress>10.198.87.19</privateIpAddress>
              <ipAddress>184.73.58.78</ipAddress>
              <sourceDestCheck>true</sourceDestCheck>
              <groupSet>
                 <item>
                     <groupId>sg-2eac845a</groupId>
                     <groupName>default</groupName>
                 </item>
              </groupSet>
              <architecture>i386</architecture>
              <rootDeviceType>ebs</rootDeviceType>
              <rootDeviceName>/dev/sda1</rootDeviceName>
              <blockDeviceMapping>
                 <item>
                   <deviceName>/dev/sda1</deviceName>
                   <ebs>
                     <volumeId>vol-a282c1cb</volumeId>
                     <status>attached</status>
                     <attachTime>2010-08-17T01:15:23.000Z</attachTime>
                     <deleteOnTermination>false</deleteOnTermination>
                   </ebs>
                 </item>
              </blockDeviceMapping>
              <instanceLifecycle>spot</instanceLifecycle>
              <spotInstanceRequestId>sir-55a3aa02</spotInstanceRequestId>
              <virtualizationType>paravirtual</virtualizationType>
              <clientToken/>
              <tagSet/>
              <hypervisor>xen</hypervisor>
          </item>
        </instancesSet>
        <requesterId>854251627541</requesterId>
      </item>
   </reservationSet>
</DescribeInstancesResponse>
```

Filtering Resources

You can reduce the results of a *describe* action by adding one or more filters to the request.

Command Line Tools

The following example uses filters to list only your volumes that are in the us-east-1b Availability Zone and have a status of `available`.

```
PROMPT>  ec2-describe-volumes --filter availability-zone=us-east-1b --filter
status=available
```

Note

If you're using the command line tools on a Windows system, you might need to use quotation marks (i.e., `--filter "status=available"`).

The following example uses filters to list only public 64-bit Windows AMIs that use an Amazon EBS volume as the root device.

```
PROMPT> ec2-describe-images --filter is-public=true --filter architecture=x86_64
 --filter platform=windows
```

The following example uses filters to list only your m1.small or m1.large instances that have an attached Amazon EBS volume that is set to delete on instance termination.

```
PROMPT> ec2-describe-instances --filter instance-type=m1.small --filter instance-
type=m1.large --filter block-device-mapping.status=attached --filter block-
device-mapping.delete-on-termination=true
```

For an example of filtering resources by tags, see Tagging Your Resources (p. 508).

For a list of the available filters for a given EC2 resource, go to the *describe* topic for that resource in the Amazon Elastic Compute Cloud Command Line Reference.

API

The following example Query request uses filters to list only your volumes that are in the us-east-1b Availability Zone and have a status of `available`.

```
https://ec2.amazonaws.com/?Action=DescribeVolumes
&Filter.1.Name=availability-zone
&Filter.1.Value.1=us-east-1b
&Filter.2.Name=status
&Filter.2.Value.1=available
&AuthParams
```

The following example Query request uses filters to list only public 64-bit Windows AMIs that use an Amazon EBS volume as the root device.

```
https://ec2.amazonaws.com/?Action=DescribeImages
&Filter.1.Name=is-public
&Filter.1.Value.1=true
&Filter.2.Name=architecture
&Filter.2.Value.1=x86_64
&Filter.3.Name=platform
```

```
&Filter.3.Value.1=windows
&AUTHPARAMS
```

The following example Query request uses filters to list only the m1.small or m1.large instances that have an attached Amazon EBS volume that is set to delete on instance termination.

```
https://ec2.amazonaws.com/?Action=DescribeInstances
&Filter.1.Name=instance-type
&Filter.1.Value.1=m1.small
&Filter.1.Value.2=m1.large
&Filter.2.Name=block-device-mapping.status
&Filter.2.Value.1=attached
&Filter.3.Name=block-device-mapping.delete-on-termination
&Filter.3.Value.1=true
&AUTHPARAMS
```

For an example of filtering resources by tags, see Tagging Your Resources (p. 508).

For a list of the available filters for a given EC2 resource, go to the *describe* topic for that resource in the Amazon Elastic Compute Cloud API Reference.

Tagging Your Resources

Topics

To help you manage your Amazon EC2 instances, images, and other Amazon EC2 resources, you can assign your own metadata to each resource in the form of *tags*. This section describes tags and how to create them.

Tag Basics

Each EC2 tag consists of a key and a value, both of which you define. For example, you could define a set of tags for your account's instances that helps you track each instance's owner and stack level. The following diagram illustrates the concept. In the diagram, you've assigned two tags to each of your instances, one called *Owner* and another called *Stack*. Each of the tags also has an associated value.

Tags let you categorize your EC2 resources in different ways, e.g., by purpose, owner, environment, etc. We recommend you devise a set of tag keys that meets your needs for each resource type. Using a consistent set of tag keys will make it easier to manage your resources. You can search and filter the resources based on the tags.

AWS doesn't apply any semantic meaning to your tags; they're interpreted strictly as strings of characters. AWS doesn't automatically set any tags on resources.

Note

You must not prefix your tag names or values with `aws:`. Tags prefixed with `aws:` have been created by AWS and cannot be edited or deleted.

You can assign tags only to resources that already exist. Exception: You can add tags when you launch an instance in the AWS Management Console. If you add a tag that has the same key as an existing tag on that resource, the new value overwrites the old value. You can edit tag keys and values, and you can remove tags from a resource at any time. You can set a tag's value to the empty string, but you can't set a tag's value to null.

If you're using AWS Identity and Access Management (IAM), you can control which Users in your AWS Account have permission to create, edit, or delete tags. For more information about IAM, see AWS Identity and Access Management (p. 9).

Tag Restrictions

The following basic restrictions apply to tags:

- **Maximum Number of Tags Per Resource—**10
- **Maximum Key Length—**128 Unicode characters
- **Maximum Value Length—**256 Unicode characters
- **Unavailable Prefix—**`aws:` (we have reserved it for tag names and values)

Tag keys and values are case sensitive.

You can't terminate, stop, or delete a resource based solely on its tags; you must specify the resource identifier instead (e.g., i-1a2b3c4d). For example, to delete snapshots that have been tagged with a tag key called `DeleteMe`, you must first get a list of the those snapshots by using a `DescribeSnapshots`

call with a filter on the tag. Then you would call `DeleteSnapshots` with the IDs of the snapshots (e.g., snap-1a2b3c4d). You can't call `DeleteSnapshots` with a filter on the tag. For more information about using filters when listing your resources, see Listing and Filtering Your Resources (p. 503).

An instance can't retrieve its own tags from its metadata (for more information about metadata, see Instance Metadata (p. 302)). Tag information is available only through the EC2 API.

Resources You Can Tag

You can tag the following Amazon EC2 and Amazon VPC resources:

- images (AMIs, kernels, RAM disks)
- instances
- security groups
- Amazon EBS volumes
- Amazon EBS snapshots
- Reserved Instances
- Spot Instance requests
- VPCs
- subnets
- Internet gateways
- VPN connections
- virtual private gateways
- customer gateways
- route tables
- network ACLs
- DHCP options sets

Following are the Amazon EC2 resources you cannot tag:

- Elastic IP addresses
- Key pairs
- Placement groups

You can tag public or shared resources, but the tags you assign are available only to your AWS account and not to the other accounts sharing the resource.

Tag Your Resources for Billing

You can use tags to organize your AWS bill to reflect your own cost structure. To do this, sign up to get your AWS account bill with tag key values included. Then, to see the cost of your combined resources, you can organize your billing information based on resources that have the same tag key values. For example, you can tag several resources with a specific application name, and then organize your billing information to see the total cost of that application across several services. For more information, see Cost Allocation and Tagging in *About AWS Account Billing*.

AWS Management Console

In this section, we'll show you how tags work in the AWS Management Console.

Note

Currently in the AWS Management Console, you can add tags only to instances, images, volumes, and snapshots.

The AWS Management Console is designed to let you add, edit, or remove tags only from a single resource at a time, whereas the EC2 API and command line tools let you do that with multiple resources in one call.

The AWS Management Console displays lists of your EC2 instances, images, and other resources. When you select a resource in one of these lists, a **Description** tab provides detailed information about the resource. Adjacent to that tab is a **Tags** tab where you can manage tags for the resource. The following image shows the tab for an instance with two tags (Name = DNS Server and Purpose = Network Management).

Adding or Deleting Tags

To add a tag, click **Add/Edit Tags**, which opens a separate dialog box like the following. The first tag on the page automatically uses *Name* as the key. However, you can change it; using *Name* is only a suggestion.

To delete a tag, click the X for the tag in the **Remove** column.

Displaying Tags as a Column in the List

Once you've added tags, you'll probably want them to appear in the resource list, so that you can sort and filter the resource list by tag. On the **Tags** tab, just click **Show Column** for the tag.

The tag appears in the resource list as a new **Name** column.

Filtering the List by Tag

You can filter your list of resources by a particular tag's value. Just click the filter icon in the top right corner of the tag's column. The following image shows filtering by *Name = DNS Server*. You can filter the list based on multiple tag keys and multiple tag values.

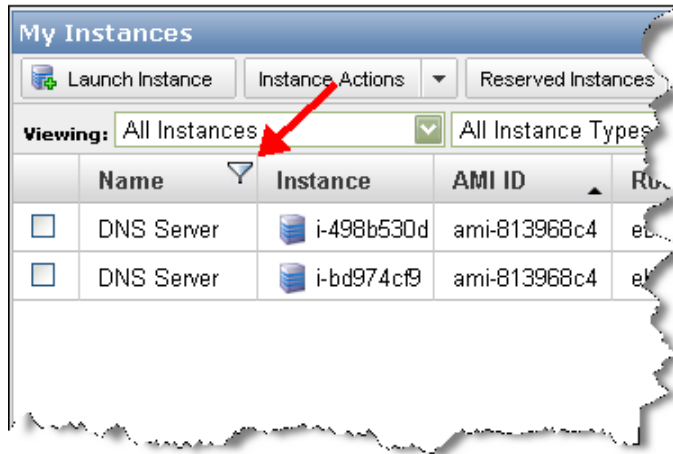When you apply the filter, the filter icon changes to indicate the filtering.

So far, you've seen how to add tags to an existing resource. The console enables you to add tags when you launch instances. There's a page in the Classic wizard where you specify tags. As mentioned previously, the first tag on the page automatically uses *Name* as the key. However, you can change it; using *Name* is only a suggestion.

Command Line Tools

For information about getting the command line tools, see Setting Up the Amazon EC2 Command Line Tools (p. 522).

Summary of available commands:

- **ec2-create-tags—**Adds a set of tags to a set of resources. You also use this to update a tag's value (overwrites the existing value).
- **ec2-delete-tags—**Deletes a set of tags from a set of resources.
- **ec2-describe-tags—**Lists your tags, or just certain ones you specify.

For details about each command, go to the Amazon Elastic Compute Cloud Command Line Reference.

Adding Tags

Let's say you want to add the same two tags to an AMI and an instance. One tag is webserver (with no value), and the other is stack=Production. In this example, the value of the webserver tag is an empty string.

```
PROMPT>  ec2-create-tags ami-1a2b3c4d i-6f5d4e3a --tag webserver --tag
stack=Production
TAG  ami-1a2b3c4d  image  webserver
TAG  ami-1a2b3c4d  image  stack  Production
TAG  i-6f5d4e3a  image  webserver
TAG  i-6f5d4e3a  image  stack  Production
```

Listing Tags

You can list your tags and filter the results different ways, as shown in the following examples.

This example describes all the tags belonging to your account.

```
PROMPT> ec2-describe-tags
TAG  ami-1a2b3c4d  image  webserver
TAG  ami-1a2b3c4d  image  stack  Production
TAG  i-5f4e3d2a  instance  webserver
TAG  i-5f4e3d2a  instance  stack  Production
TAG  i-12345678  instance  database_server
TAG  i-12345678  instance  stack  Test
```

This example describes the tags for the AMI with ID ami-1a2b3c4d.

```
PROMPT> ec2-describe-tags --filter "resource-id=ami-1a2b3c4d"
TAG  ami-1a2b3c4d  image  webserver
TAG  ami-1a2b3c4d  image  stack  Production
```

This example describes the tags for all your instances.

```
PROMPT> ec2-describe-tags --filter "resource-type=instance"
TAG  i-5f4e3d2a  instance  webserver
TAG  i-5f4e3d2a  instance  stack  Production
TAG  i-12345678  instance  database_server
TAG  i-12345678  instance  stack  Test
```

This example describes the tags for all your instances tagged with the name *webserver*.

```
PROMPT> ec2-describe-tags --filter "resource-type=instance" --filter "key=web
server"
TAG  i-5f4e3d2a  instance  webserver
```

This example describes the tags for all your instances tagged with either stack=Test or stack=Production.

```
PROMPT> ec2-describe-tags --filter "resource-type=instance" --filter "key=stack"
 --filter "value=Test" --filter "value=Production"
TAG  i-5f4e3d2a  instance  stack  Production
TAG  i-12345678  instance  stack  Test
```

This example describes the tags for all your instances tagged with Purpose=[empty string].

```
PROMPT> ec2-describe-tags --filter "resource-type=instance" --filter "key=Pur
pose" --filter "value="
```

Changing Tag Values

Let's say you now want to change the value of the stack tag for one of the AMIs from Production to Test. You use the ec2-create-tags command to overwrite the original tag value.

```
PROMPT>  ec2-create-tags ami-1a2b3c4d  --tag stack=Test
TAG  ami-1a2b3c4d  image  stack  Test
```

Deleting Tags

Now let's say you want to delete the tags you originally assigned to the AMI and instance. You don't need to specify the value.

```
PROMPT>  ec2-delete-tags ami-1a2b3c4d i-6f5d4e3a --tag webserver --tag stack
```

If you specify a value for the key, the tag is deleted only if the tag's value matches the one you specified. If you specify the empty string as the value, the tag is deleted only if the tag's value is the empty string. The following example specifies the empty string as the value for the tag to delete (notice the equals sign after Owner).

```
PROMPT>  ec2-delete-tags snap-1a2b3c4d --tag Owner=
```

Filtering the List of Resources by Tags

You can describe your resources and filter the results based on the tag. Let's say that you've tagged all your Amazon EBS volumes with an Owner tag and a Purpose tag. You've got a series of teams (TeamA, TeamB, TeamC, etc.), and each has a series of volumes they own. You can get tag descriptions and filter the results by volume.

```
PROMPT>  ec2-describe-tags --filter resource-type=volume
TAG  vol-abcd1234  volume  Owner  TeamA
TAG  vol-abcd1234  volume  Purpose  Project1
TAG  vol-efba9876  volume  Owner  TeamA
TAG  vol-efba9876  volume  Purpose Project2
```

```
TAG   vol-4562dabf   volume   Owner   TeamA
TAG   vol-4562dabf   volume   Purpose   RawLogData
TAG   vol-2a3d4b5f   volume   Owner   TeamB
TAG   vol-2a3d4b5f   volume   Purpose Project1
TAG   vol-9f8g7d6c   volume   Owner   TeamB
TAG   vol-9f8g7d6c   volume   Purpose   Project2
TAG   vol-3b3a4c4d   volume   Owner   TeamB
TAG   vol-3b3a4c4d   volume   Purpose Logs
TAG   vol-1234abcd   volume   Owner   TeamC
TAG   vol-1234abcd   volume   Purpose Project1
TAG   vol-7f7g7d7a   volume   Owner   TeamC
TAG   vol-7f7g7d7a   volume   Purpose   Project2
TAG   vol-4a4b4c4d   volume   Owner   TeamC
TAG   vol-4a4b4c4d   volume   Purpose Logs
```

Likewise, you can get volume descriptions, and filter the results by tag. The filter name you use is `tag:`*`key`*. Let's say you want to get a list of just the volumes belonging to either TeamA or TeamB, and that contain log data. This time you use `ec2-describe-volumes` with a filter based on the tags of interest. You use a wildcard to find the volumes with "Log" in the *Purpose* tag's value.

```
PROMPT>  ec2-describe-volumes --filter tag:Owner=TeamA --filter tag:Owner=TeamB
 --filter tag:Purpose=*Log*
VOLUME   vol-4562dabf   5   us-east-1b   available    2010-02-22T22:50:43+0000
Owner   TeamA   Purpose   RawLogData
VOLUME    vol-3b3a4c4d   12   us-east-1b   available   2010-05-01T13:09:27+0000
Owner    TeamB   Purpose Logs
```

API

Summary of available API actions:

- **CreateTags—**Adds a set of tags to a set of resources. You also use this to update a tag's value (overwrites the existing value).
- **DeleteTags—**Deletes a set of tags from a set of resources.
- **DescribeTags—**Lists your tags, or just certain ones you specify.

For details about each action and examples, go to the Amazon Elastic Compute Cloud API Reference.

Using the API to add, update, list, and delete tags is straightforward. The topics in the API reference show examples. In the following sections, we will demonstrate how you can list tags, or list resources and filter the results based on tags.

Filtering the List of Resources by Tags

You can describe your resources and filter the results based on the tag. Let's say that you've tagged all your Amazon EBS volumes with an `Owner` tag and a `Purpose` tag. You've got a series of teams (TeamA, TeamB, TeamC, etc.), and each has a series of volumes they own. You can get tag descriptions and filter the results by volume.

Example Query request:

```
https://ec2.amazonaws.com/?Action=DescribeTags
&Filter.1.Name=resource-type
```

```
&Filter.1.Value=volume
&AuthParams
```

In this example, the response includes 18 tags covering 9 volumes.

```
<DescribeTagsResponse xmlns="http://ec2.amazonaws.com/doc/2012-07-20/">
    <requestId>7a62c49f-347e-4fc4-9331-6e8eEXAMPLE</requestId>
    <tagSet>
        <item>
            <resourceId>vol-abcd1234</resourceId>
            <resourceType>volume</resourceType>
            <key>Owner</key>
            <value>TeamA</value>
        </item>
        <item>
            <resourceId>vol-abcd1234</resourceId>
            <resourceType>volume</resourceType>
            <key>Purpose</key>
            <value>Project1</value>
        </item>
        <item>
            <resourceId>vol-efba9876</resourceId>
            <resourceType>volume</resourceType>
            <key>Owner</key>
            <value>TeamA</value>
        </item>
        <item>
            <resourceId>vol-efba9876</resourceId>
            <resourceType>volume</resourceType>
            <key>Purpose</key>
            <value>Project2</value>
        </item>
        <item>
            <resourceId>vol-4562dabf</resourceId>
            <resourceType>volume</resourceType>
            <key>Owner</key>
            <value>TeamA</value>
        </item>
        <item>
            <resourceId>vol-4562dabf</resourceId>
            <resourceType>volume</resourceType>
            <key>Purpose</key>
            <value>RawLogData</value>
        </item>
        <item>
            <resourceId>vol-2a3d4b5f</resourceId>
            <resourceType>volume</resourceType>
            <key>Owner</key>
            <value>TeamB</value>
        </item>
        <item>
            <resourceId>vol-2a3d4b5f</resourceId>
            <resourceType>volume</resourceType>
            <key>Purpose</key>
            <value>Project1</value>
        </item>
        <item>
```

```
    <resourceId>vol-9f8g7d6c</resourceId>
    <resourceType>volume</resourceType>
    <key>Owner</key>
    <value>TeamB</value>
</item>
<item>
    <resourceId>vol-9f8g7d6c</resourceId>
    <resourceType>volume</resourceType>
    <key>Purpose</key>
    <value>Project2</value>
</item>
<item>
    <resourceId>vol-3b3a4c4d</resourceId>
    <resourceType>volume</resourceType>
    <key>Owner</key>
    <value>TeamB</value>
</item>
<item>
    <resourceId>vol-3b3a4c4d</resourceId>
    <resourceType>volume</resourceType>
    <key>Purpose</key>
    <value>Logs</value>
</item>
  <item>
    <resourceId>vol-1234abcd</resourceId>
    <resourceType>volume</resourceType>
    <key>Owner</key>
    <value>TeamC</value>
</item>
<item>
    <resourceId>vol-1234abcd</resourceId>
    <resourceType>volume</resourceType>
    <key>Purpose</key>
    <value>Project1</value>
</item>
<item>
    <resourceId>vol-7f7g7d7a</resourceId>
    <resourceType>volume</resourceType>
    <key>Owner</key>
    <value>TeamC</value>
</item>
<item>
    <resourceId>vol-7f7g7d7a</resourceId>
    <resourceType>volume</resourceType>
    <key>Purpose</key>
    <value>Project2</value>
</item>
<item>
    <resourceId>vol-4a4b4c4d</resourceId>
    <resourceType>volume</resourceType>
    <key>Owner</key>
    <value>TeamC</value>
</item>
<item>
    <resourceId>vol-4a4b4c4d</resourceId>
    <resourceType>volume</resourceType>
    <key>Purpose</key>
    <value>Logs</value>
```

```
        </item>
    </tagSet>
</DescribeTagsResponse>
```

Likewise, you can get volume descriptions, and filter the results by tag. The filter name you use is `tag:`*key*. Let's say you want to get a list of just the volumes belonging to either TeamA or TeamB, and that contain log data. This time you use `DescribeVolumes` with a filter based on the tags of interest. You use a wildcard to find the volumes with "Log" in the `Purpose` tag's value.

Example Query request:

```
https://ec2.amazonaws.com/?Action=DescribeVolumes
&Filter.1.Name=tag:Owner
&Filter.1.Value.1=TeamA
&Filter.1.Value.2=TeamB
&Filter.2.Name=tag:Purpose
&Filter.2.Value.1=*Log*
&AuthParams
```

Because of the filtering, the response includes only two of the volumes that were in the preceding list of tagged volumes. The volume's tags are included in the response.

```
<DescribeVolumesResponse xmlns="http://ec2.amazonaws.com/doc/2012-07-20/">
   <requestId>7a62c49f-347e-4fc4-9331-6e8eEXAMPLE</requestId>
   <volumeSet>
      <item>
         <volumeId>vol-4562dabf</volumeId>
         <size>5</size>
         <snapshotId/>
         <availabilityZone>us-east-1b</availabilityZone>
         <status>available</status>
         <createTime>2010-02-22T22:50:43+0000</createTime>
         <attachmentSet/>
         <tagSet>
            <item>
               <key>Owner</key>
               <value>TeamA</key>
            </item>
            <item>
               <key>Purpose</key>
               <value>RawLogData</key>
            </item>
         </tagSet>
      </item>
      <item>
         <volumeId>vol-3b3a4c4d</volumeId>
         <size>12</size>
         <snapshotId/>
         <availabilityZone>us-east-1b</availabilityZone>
         <status>available</status>
         <createTime>2010-05-01T13:09:27+0000</createTime>
         <attachmentSet/>
         <tagSet>
            <item>
               <key>Owner</key>
               <value>TeamB</key>
```

```
                    </item>
                     <item>
                        <key>Purpose</key>
                        <value>Logs</key>
                    </item>
                </tagSet>
            </item>
        </volumeSet>
</DescribeVolumesResponse>
```

Setting Up the Amazon EC2 Command Line Tools

The Amazon EC2 command line tools (also called the *API tools* or the *CLI tools*) wrap the Amazon EC2 API actions. These tools are written in Java and include shell scripts for both Windows and Linux/UNIX/Mac OSX.

Before you can use the Amazon EC2 command line tools, you need to download them and configure them to use your AWS account.

Complete the following tasks to set up your Amazon EC2 environment:

Topics

For a detailed reference guide that describes the commands, see the Amazon Elastic Compute Cloud Command Line Reference.

Download the Command Line Tools (API Tools)

The command line tools are available as a ZIP file on this site: Amazon EC2 API Tools. The ZIP file is self-contained; no installation is required. You can simply download the file and unzip it.

Optional step before you unzip: For detailed information about authenticating the download before unzipping the file, see Verify the Signature of the Tools Download (p. 523).

Some additional setup is required before you can use the tools. These steps are discussed next.

Verify the Signature of the Tools Download

Whenever you download an application from the Internet, you should authenticate the identity of the software publisher and check that the application has not been altered or corrupted since it was published. This protects you from installing a version of the application that contains a virus or other malicious code.

If you determine that the software for the command line tools has been altered, do not unzip or install the file that you downloaded. Instead, contact Amazon Web Services.

Topics

Overview

The first step is to establish trust with the software publisher: download the public key of the software publisher, check that the owner of the public key is who they claim to be, and then add the public key to your keyring. Your keyring is a collection of known public keys. You can then explicitly trust the public key, or trust it implicitly because the public key is trusted by someone with whom you have a pre-existing trust relationship.

After you've established the authenticity of the public key, you can use it to verify the signature of the application. Using security tools, you'll calculate a signature from the publisher's public key and your downloaded copy of the application. If the calculated signature matches the signature the software developer has published for the application, you can have confidence that the application has not been altered.

Amazon EC2 command line tools are signed using GnuPG, an open implementation of the Pretty Good Privacy (OpenPGP) standard for secure digital signatures. GnuPG provides authentication and integrity checking through a 128-bit digital signature. Amazon EC2 publishes a public key and signatures you can use to verify the downloaded Amazon EC2 command line tools. For more information about PGP and GnuPG (GPG), go to http://www.gnupg.org.

Install the GPG Tools

If your operating system is Linux or UNIX, the GPG tools are likely already installed. To test whether the tools are installed on your system, type `gpg` at a command-line prompt. If the GPG tools are installed, you get a GPG command prompt. If the GPG tools are not installed, you get an error stating that the command cannot be found. You can install the GnuPG package from a repository.

To install GPG Tools on Debian-based Linux

- From a terminal, run the following command.

```
apt-get install gnupg
```

To install GPG Tools on Red Hat-based Linux

- From a terminal, run the following command.

```
yum install gnupg
```

To install GPG Tools on Windows

*   Download and install Gpg4win, an implementation of GnuPG that runs on Windows.

    During installation of Gpg4win, you can use the default values suggested by Gpg4win. As part of the installation process, you are asked whether you want to define a root certificate. Defining a root certificate is a way to establish trust with many software publishers; setting a root certificate establishes trust with all publishers trusted by that certificate. If you want to define a root certificate, follow the instructions in the text box. If not, click the chekbox to skip this step. Either option is fine for the purpose of verifying the signature of the Amazon EC2 API tools package.

To install GPG Tools on Mac OS

*   Download and install GPGTools, an implementation of GnuPG that runs on Mac OS.

In addition to installing pre-compiled implementations of GnuPG, you can also download and compile the source code from http://gnupg.org/download/index.en.html.

After you have installed a set of GPG tools, use them to create a public-private key set. You'll need this later to sign changes to your trust status. If you have previously installed the GPG tools and already have a public-private key set, you can skip this step.

To create a private key for the GPG tools

1. From the command line, run the following command.

```
gpg --gen-key
```

2. Answer the questions that follow. You can use the suggested default values. Make a note of the passphrase you use to create the private key. You'll need this value later.

Authenticate the Public Key

The next step in the process is to authenticate the EC2 Packages public key and add it as a trusted key in your GPG keyring.

To authenticate the EC2 Packages public key

1. Create a text file named `ec2-packages-public.key` and copy the public key from EC2 Packages Public Key into the text file. This includes everything from `-----BEGIN PGP PUBLIC KEY BLOCK-----` to `-----END PGP PUBLIC KEY BLOCK-----`. Save the text file.
2. Import the EC2 Packages public key into your keyring using the following command in the directory where you saved the file `ec2-packages-public.key`.

```
gpg --import ec2-packages-public.key
```

    The command returns results similar to the following. Make a note of the key value; you'll need it in the next step. In the example below, the key value is `0349E66A`.

```
pg: key 0349E66A: public key "AWS EC2 Packages <ec2-packages@amazon.com>"
imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0  valid:   2  signed:   1  trust: 0-, 0q, 0n, 0m, 0f, 2u
gpg: depth: 1  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 1f, 0u
gpg: next trustdb check due at 2014-07-20
```

3. Verify the fingerprint by running the following command, where *key-value* is replaced by the value from the previous step.

```
gpg --fingerprint key-value
```

This command returns results similar to the following. Compare the key fingerprint to that published on EC2 Packages Public Key. They should match. If they don't, do not continue to install the tools, and contact Amazon Web Services.

```
pub   4096R/0349E66A 2011-04-04
Key fingerprint = A262 37CF 2294 C30E 9844  96C9 116B 3651 0349 E66A
uid   AWS EC2 Packages <ec2-packages@amazon.com>
```

4. If the fingerprint matches the one published on the aws.amazon.com website, you may choose to trust EC2 Packages public key. To do so, run the following command, where *key-value* is replaced by the key value from Step 2.

```
gpg --edit-key key-value
```

Type the following command at the GPG tools prompt.

```
trust
```

The GPG tools ask you to establish a level of trust for EC2 Packages, with a prompt such as the following. To learn more about these trust options, go to the The GNU Privacy Handbook.

```
Please decide how far you trust this user to correctly verify other users'
 keys
(by looking at passports, checking fingerprints from different sources,
etc.)

  1 = I don't know or won't say
  2 = I do NOT trust
  3 = I trust marginally
  4 = I trust fully
  5 = I trust ultimately
  m = back to the main menu

Your decision?
```

Type 4 and press the Enter key.

5. Sign the key with your private key (created when you installed the GPG tools) to set the new trust level. Do this using the following command.

```
sign
```

You are asked to confirm, type `y`, and press the Enter key. You are asked for the passcode that you used when you created your private key. Type your passcode and press the Enter key.

6. Save your changes using the following command.

```
save
```

This saves your changes to the keyring. It should also exit the PGP session. If it doesn't, press CTRL+Z to exit the PGP session and return to the main terminal.

Verify the Signature of the Package

With the GPG tools installed and the EC2 Packages public key authenticated and the EC2 Packages public key trusted, you're ready to check the signature of the Amazon EC2 API tools package.

To verify the signature of Amazon EC2 API tools package

1. Download the Amazon EC2 command line tools, `ec2-api-tools.zip`, from the Amazon EC2 API Tools.
2. Create a new text file name `ec2-api-tools.zip.asc` and copy the contents of the Amazon EC2 command line tools signature into this file. Copy everything including the `-----BEGIN PGP SIGNATURE-----` to `-----END PGP SIGNATURE-----` lines. Save the file.
3. Verify the signature of the command line tools by typing the following at a command line prompt in the directory where you saved the file `ec2-api-tools.zip.asc` and the command line tool ZIP file `ec2-api-tools.zip`. Both files must be present.

```
gpg --verify ec2-api-tools.zip.asc ec2-api-tools.zip
```

The output should be something like the following.

```
gpg: Signature made Mon Mar 12 14:51:33 2012 PDT using RSA key ID 0349E66A
gpg: Good signature from "AWS EC2 Packages <ec2-packages@amazon.com>"
```

If the output contains the phrase 'Good signature from "AWS EC2 Packages <ec2-packages@amazon.com>"' the signature has successfully been verified, and you can proceed to unzip and install the Amazon EC2 command line tools. If the output includes the phrase "BAD signature", check that you performed the procedure correctly. If you continue to get this response, contact Amazon Web Services and do not unzip or install the file that you downloaded.

Tell the Tools Where Java Lives

The Amazon EC2 command line tools require Java. If you don't have Java 1.6 or later installed, download and install Java. Either a JRE or JDK installation is acceptable. To view and download JREs for a range of platforms, including Linux/UNIX and Windows, go to http://java.oracle.com.

The Amazon EC2 command line tools read the `JAVA_HOME` environment variable to locate the Java runtime. This environment variable should specify the full path of the directory that contains a subdirectory named `bin` that contains the Java executable you installed (`java` on Linux and UNIX, `java.exe` on Windows).

On Linux and UNIX, you can set the JAVA_HOME environment variable as follows.

To set the JAVA_HOME environment variable on Linux/UNIX

1. You can verify whether you have Java installed and where it is located by using the following command:

```
$ which java
```

The following is example output.

```
/usr/bin/java
```

2. Set JAVA_HOME to the full path of the Java home directory. For example, if your Java executable is in /user/bin, set JAVA_HOME to /usr, as shown here.

```
$ export JAVA_HOME=/usr
```

> **Note**
>
> The export command applies only to the current shell session. To permanently create or update an environment variable, include the command in a start-up script. For example, if you use Bash shell, you can include commands in your ~/.bashrc or /etc/profile file.

> **Note**
>
> If you are using Cygwin, JAVA_HOME should have a Windows path.

3. You can verify your JAVA_HOME setting using this command.

```
$ $JAVA_HOME/bin/java -version
```

If you've set the environment variable correctly, the output looks something like this.

```
java version "1.7.0_05"
Java(TM) SE Runtime Environment (build 1.7.0_05-b05)
Java HotSpot(TM) Client VM (build 23.1-b03, mixed mode, sharing)
```

On Windows, you can set the JAVA_HOME environment variable as follows.

To set the JAVA_HOME environment variable on Windows

1. Set JAVA_HOME to the full path of the Java home directory. For example, if your Java executable is in C:\jdk\bin, set JAVA_HOME to C:\jdk.

```
C:\> setx JAVA_HOME=C:\jdk
```

> **Note**
>
> If the path contains spaces, enclose it in quotes. For example, "C:\Program Files (x86)\Java\jre7".

2. You can verify your JAVA_HOME setting using this command:

```
C:\> %JAVA_HOME%\bin\java -version
```

If you've set the environment variable correctly, the output looks something like this:

```
java version "1.7.0_05"
Java(TM) SE Runtime Environment (build 1.7.0_05-b05)
Java HotSpot(TM) Client VM (build 23.1-b03, mixed mode, sharing)
```

Otherwise, check the setting of JAVA_HOME using the **set** command, fix any errors, and try the command again.

3.   Add the `bin` directory that contains the Java exectuable to your path before other versions of Java.

```
C:\> setx Path=%JAVA_HOME%\bin;"%Path%
```

4.   You can verify your update to the `Path` environment variable using this command:

```
C:\> java -version
```

You should see the same output as before. Otherwise, check the setting of `Path`, fix any errors, and try the command again.

Tell the Tools Where They Live

The command line tools read the `EC2_HOME` environment variable to locate supporting libraries. Before using these tools, set `EC2_HOME` to the directory path where you unzipped the command line tools. This directory is named `ec2-api-tools-`*w.x.y.z* (where *w*, *x*, *y*, and *z* are components of the version number). It contains sub-directories named `bin` and `lib`.

In addition, to make things a little easier, you can add the `bin` directory for the tools to your system path. The examples in the EC2 User Guide assume that you have done so.

On Linux and UNIX, you can set these environment variables as follows. This **export** command updates the current shell session only. To set an environment variable persistently, include this command in the start-up script for your shell.

To set the EC2_HOME and PATH environment variable on Linux/UNIX

1.   Use this command to set the `EC2_HOME` environment variable.

```
$ export EC2_HOME=<path-to-tools>
```

> **Note**
>
> If you are using Cygwin, `EC2_HOME` must use Linux/UNIX paths (for example, `/usr/bin` instead of `C:\usr\bin`). Additionally, the value of `EC2_HOME` cannot contain any spaces, even if the value is quoted or the spaces are escaped.

2.   You can update your `PATH` as follows.

```
$ export PATH=$PATH:$EC2_HOME/bin
```

On Windows, you can set these environment variables as follows.

To set the `EC2_HOME` and `Path` environment variables on Windows

1. Use this command to set the `EC2_HOME` environment variable.

    ```
    C:\> setx EC2_HOME=<path-to-tools>
    ```

2. You can verify your `EC2_HOME` setting using this command.

    ```
    C:\> dir %EC2_HOME%
    ```

    If you've set the environment variable correctly, you'll see output for the directory listing. If you get a File Not Found error, check the setting of `EC2_HOME`, fix any errors, and try the command again.

3. You can update your `Path` as follows.

    ```
    C:\> setx Path=%PATH%;%EC2_HOME%\bin
    ```

Tell the Tools Who You Are

When you sign up for an AWS account, we create access credentials for you so that you can make secure requests to AWS. You must provide these credentials to the Amazon EC2 command line tools so that they know that the commands that you issue come from your account.

The following procedure describes how you can view your access credentials.

To view your AWS access credentials

1. Go to the Amazon Web Services website at http://aws.amazon.com.
2. Click **My Account/Console**, and then click **Security Credentials**.
3. Under **Your Account**, click **Security Credentials**.
4. In the spaces provided, type your user name and password, and then click **Sign in using our secure server**.
5. Under **Access Credentials**, on the **Access Keys** tab, your access key ID is displayed. To view your secret key, under **Secret Access Key**, click **Show**.

You can specify these credentials with the **--aws-access-key** and **--aws-secret-key** (or **-O** and **-W**) options every time you issue a command. However, it's easier to specify your access credentials using the following environment variables:

*   `AWS_ACCESS_KEY`—Your access key ID
*   `AWS_SECRET_KEY`—Your secret access key

If these environment variables are set properly, their values serve as the default values for these required options, so you can omit them from the command line.

Note

Although we don't encourage it, for a limited time you can still use `EC2_PRIVATE_KEY` and `EC2_CERT` instead of `AWS_ACCESS_KEY` and `AWS_SECRET_KEY`. For more information, see *Deprecated Options* in Common Options for API Tools in the *Amazon Elastic Compute Cloud CLI Reference.* If you specify both sets of credentials, the command line tools use the access key ID and secret access key.

On Linux and UNIX, you can set these environment variables as follows. These commands update the current shell session only. To permanently set these environment variables, include these commands in the start-up script for your shell.

```
$ export AWS_ACCESS_KEY=your_AWS_ACCESS_KEY_ID
$ export AWS_SECRET_KEY=your_AWS_SECRET_KEY
```

On Windows, you can set these environment variables as follows.

```
C:\>  setx AWS_ACCESS_KEY=your_AWS_ACCESS_KEY_ID
C:\> setx AWS_SECRET_KEY=your_AWS_SECRET_KEY
```

Set the Region

Amazon EC2 prices vary by Region. By default, the Amazon EC2 tools use the US East (Northern Virginia) Region (`us-east-1`) with the `ec2.us-east-1.amazonaws.com` service endpoint URL. If your instances are in a different region, you must specify the region where your instances reside. This section shows how to specify a different Region by setting the service endpoint URL.

To set the service endpoint URL

1. To view the available Regions, use the following Amazon EC2 command: ec2-describe-regions.

```
PROMPT> ec2-describe-regions
REGION   ap-northeast-1        ec2.ap-northeast-1.amazonaws.com
REGION   ap-southeast-1        ec2.ap-southeast-1.amazonaws.com
..
```

The result shows the Region names and corresponding service endpoints.

Note

If you get an error that required option **-O** is missing, check the setting of `AWS_ACCESS_KEY`, fix any errors, and try the command again.

If you get an error that required option **-W** is missing, check the setting of `AWS_SECRET_KEY`, fix any errors, and try the command again.

If you get an error that required option **-C** is missing, check the setting of `EC2_CERT`, fix any errors, and try the command again.

If you get an error that required option **-K** is missing, check the setting of `EC2_PRIVATE_KEY`, fix any errors, and try the command again.

2. Set the `EC2_URL` environment variable using the service endpoint URL returned in the previous step.

For Linux and UNIX, use the following syntax. The **export** command updates the current shell session only. To set an environment variable persistently, include this command in the start-up script for your shell.

```
$ export EC2_URL=https://<service_endpoint>
```

For Windows, use the following syntax.

```
C:\> setx EC2_URL=https://<service_endpoint>
```

For information about pricing, see Amazon EC2 Pricing.

Download an SSH Client (Linux/UNIX)

For some of the examples in the EC2 User Guide, you'll need an SSH client. Most Linux and UNIX installations include an SSH client by default. If yours doesn't, the OpenSSH project provides a free implementation of the full suite of SSH tools. For more information, go to http://www.openssh.org.

Windows users can download and install PuTTY, a free SSH client. To download the client and installation instructions, go to http://www.chiark.greenend.org.uk/~sgtatham/putty/. For information on how to use PuTTY with Amazon EC2, see Connecting to Linux/UNIX Instances from Windows Using PuTTY (p. 269).

Download a Remote Desktop Client (Windows)

For some of the examples in the EC2 user Guide, you'll need a Remote Desktop client. The most recent versions of Windows include a Remote Desktop client by default. To check whether you have one, open a Command Prompt window and type **mstsc**. If this command displays the Remote Desktop Connection window, you're set. Otherwise, go to the Microsoft Windows home page and search for the download for Remote Desktop Connection.

Making API Requests

Topics

This section provide an introduction to using the Query and SOAP APIs for EC2.

Required Knowledge

If you plan to access EC2 through its API, we assume that you're familiar with the following:

- XML
- Web services
- HTTP requests
- One or more programming languages, such as Java, PHP, Perl, Python, Ruby, C#, or C++.

Endpoints

For information about this product's regions and endpoints, go to Regions and Endpoints in the Amazon Web Services General Reference.

If you just specify the general endpoint (ec2.amazonaws.com), the us-east-1 endpoint is used. If you want to use a different region, you need to specify the endpoint associated with it. For example, if you specify ec2.us-west-2.amazonaws.com as the endpoint, Amazon EC2 directs your request to the us-west-2 endpoint which is used for the US West (Oregon) region.

For more information about Regions, see Region and Availability Zone Concepts (p. 111).

Making Query Requests

Topics

Query requests are HTTP or HTTPS requests that use the HTTP verb GET or POST and a Query parameter named *Action*.

Endpoints

For information about this product's regions and endpoints, go to Regions and Endpoints in the Amazon Web Services General Reference.

If you just specify the general endpoint (ec2.amazonaws.com), the us-east-1 endpoint is used.

For more information about Regions, see Region and Availability Zone Concepts (p. 111).

Structure of a GET Request

This guide presents the EC2 GET requests as URLs, which can be used directly in a browser. The URL consists of:

- **Endpoint—**The web service entry point to act on (e.g., ec2.amazonaws.com, which defaults to the us-east-1 Region)
- **Action—**The action you want to perform on the endpoint; for example: running an instance (`RunInstance`)
- **Parameters—**Any request parameters

The following is an example GET request to run instances.

```
https://ec2.amazonaws.com/?Action=RunInstances&ImageId=ami-60a54009&Max
Count=3&MinCount=1&Placement.AvailabilityZone=us-east-1b&Monitoring.En
abled=true&AWSAccessKeyId=0GS7553JW74RRM612K02EXAMPLE&Version=2012-07-20&Ex
pires=2010-10-10T12:00:00Z&Signature=lBP67vCvGlDMBQ1dofZxg8E8SUEXAMPLE&Signa
tureVersion=2&SignatureMethod=HmacSHA256
```

Important

Because the GET requests are URLs, you must URL encode the parameter values. In the EC2 documentation, we leave the example GET requests unencoded to make them easier to read.

To make the GET examples even easier to read, this guide presents them in the following parsed format.

```
https://ec2.amazonaws.com/
?Action=RunInstances
&ImageId=ami-60a54009
&MaxCount=3
&MinCount=1
```

```
&Placement.AvailabilityZone=us-east-1b
&Monitoring.Enabled=true
&AWSAccessKeyId=0GS7553JW74RRM612K02EXAMPLE
&Version=2012-07-20
&Expires=2010-10-10T12:00:00Z
&Signature=lBP67vCvGlDMBQldofZxg8E8SUEXAMPLE
&SignatureVersion=2
&SignatureMethod=HmacSHA256
```

The first line represents the *endpoint* of the request. After the endpoint is a question mark (?), which separates the endpoint from the parameters. Each parameter is separated by an ampersand (&).

The *Action* parameter indicates the action to perform (for a list of the actions, go to Amazon Elastic Compute Cloud API Reference).

Note

In the example Query requests we present in the EC2 documentation, we omit the parameters related to authentication to make it easier to focus on the ones relevant to the particular operation. We replace them with the following literal string to remind you that a real request includes the parameters: &AuthParams.

Query Parameters

Each Query request must include some common parameters to handle authentication and selection of an action. For more information, go to the Amazon Elastic Compute Cloud API Reference.

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

You can send Query requests over either HTTP or HTTPS. Regardless of which protocol you use, you must include a signature in every Query request. This section describes how to create the signature. The method described in the following procedure is known as *signature version 2*.

Caution

If you are currently using signature version 1: Version 1 is deprecated, and you should move to signature version 2 immediately. For information about the deprecation schedule and the differences between signature version 2 and version 1, go to Making Secure Requests to Amazon Web Services.

To create the signature

1. Create the canonicalized query string that you need later in this procedure:

    a. Sort the UTF-8 query string components by parameter name with natural byte ordering. The parameters can come from the GET URI or from the POST body (when Content-Type is application/x-www-form-urlencoded).

    b. URL encode the parameter name and values according to the following rules:

        *   Do not URL encode any of the unreserved characters that RFC 3986 defines. These unreserved characters are A-Z, a-z, 0-9, hyphen (-), underscore (_), period (.), and tilde (~).

- Percent encode all other characters with %XY, where X and Y are hex characters 0-9 and uppercase A-F.
- Percent encode extended UTF-8 characters in the form %XY%ZA....
- Percent encode the space character as %20 (and not +, as common encoding schemes do).

> **Note**
>
> Currently all AWS service parameter names use unreserved characters, so you don't need to encode them. However, you might want to include code to handle parameter names that use reserved characters, for possible future use.

    c. Separate the encoded parameter names from their encoded values with the equals sign (=) (ASCII character 61), even if the parameter value is empty.

    d. Separate the name-value pairs with an ampersand (&) (ASCII code 38).

2. Create the string to sign according to the following pseudo-grammar (the `"\n"` represents an ASCII newline).

```
StringToSign = HTTPVerb + "\n" +
               ValueOfHostHeaderInLowercase + "\n" +
               HTTPRequestURI + "\n" +
               CanonicalizedQueryString <from the preceding step>
```

The HTTPRequestURI component is the HTTP absolute path component of the URI up to, but not including, the query string. If the HTTPRequestURI is empty, use a forward slash (/).

3. Calculate an RFC 2104-compliant HMAC with the string you just created, your Secret Access Key as the key, and SHA256 or SHA1 as the hash algorithm.
   For more information, go to http://www.ietf.org/rfc/rfc2104.txt.

4. Convert the resulting value to base64.

5. Use the resulting value as the value of the *Signature* request parameter.

> **Important**
>
> The final signature you send in the request must be URL encoded as specified in RFC 3986 (for more information, go to http://www.ietf.org/rfc/rfc3986.txt). If your toolkit URL encodes your final request, then it handles the required URL encoding of the signature. If your toolkit doesn't URL encode the final request, then make sure to URL encode the signature before you include it in the request. Most importantly, make sure the signature is URL encoded *only once*. A common mistake is to URL encode it manually during signature formation, and then again when the toolkit URL encodes the entire request.

Example DescribeImages Request

```
https://ec2.amazonaws.com/
?Action=DescribeImages
&ImageId.1=ami-2bb65342
&Version=2012-07-20
&Expires=2008-02-10T12%3A00%3A00Z
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&AWSAccessKeyId=<Your AWS Access Key ID>
```

Following is the string to sign.

```
GET\n
ec2.amazonaws.com\n
/\n
AWSAccessKeyId=<Your AWS Access Key ID>
&Action=DescribeImages
&Expires=2008-02-10T12%3A00%3A00Z
&ImageId.1=ami-2bb65342
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Version=2012-07-20
```

Following is the signed request.

```
https://ec2.amazonaws.com/
?Action=DescribeImages
&ImageId.1=ami-2bb65342
&Version=2012-07-20
&Expires=2008-02-10T12%3A00%3A00Z
&Signature=<URLEncode(Base64Encode(Signature))>
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&AWSAccessKeyId=<Your AWS Access Key ID>
```

Making SOAP Requests

Topics

- Endpoints (p. 536)
- WSDL and Schema Definitions (p. 537)
- Programming Language Support in Amazon EC2 (p. 537)
- Request Authentication (p. 537)

Endpoints

For information about this product's regions and endpoints, go to Regions and Endpoints in the Amazon Web Services General Reference. If you just specify the general endpoint (ec2.amazonaws.com), the us-east-1 endpoint is used. For more information about Regions, see Region and Availability Zone Concepts (p. 111).

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.

All schemas have a version number (the latest is 2012-07-20). 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://s3.amazonaws.com/ec2-downloads/ec2.wsdl'. At the time this document was released, the current API version was 2012-07-20.

The following are additional web service references.

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

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

To prevent in-flight tampering, all SOAP requests should be sent over HTTPS. In addition, the 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/2012-07-20/">
    <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-wsse
curity-utility-1.0.xsd"
      EncodingType="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 Algorithm="http://www.w3.org/2001/10/xml-
exc-c14n#"></ds:CanonicalizationMethod>
          <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-
sha1"></ds:SignatureMethod>
          <ds:Reference URI="#id-17984263">
            <ds:Transforms>
              <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#"></ds:Transform>
            </ds:Transforms>
            <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmld
sig#sha1"></ds:DigestMethod>
            <ds:DigestValue>0pjZ1+TvgPf6uG7o+Yp3l2YdGZ4=</ds:DigestValue>
          </ds:Reference>
          <ds:Reference URI="#id-15778003">
            <ds:Transforms>
              <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#"></ds:Transform>
            </ds:Transforms>
            <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmld
sig#sha1"></ds:DigestMethod>
            <ds:DigestValue>HhRbxBBmc2OO348f8nLNZyo4AOM=</ds:DigestValue>
          </ds:Reference>
        </ds:SignedInfo>
      <ds:SignatureValue>bmVx24Qom4kd9QQtclxWIlgLk4QsQBPaKESi79x479xgbO9PEStXMi
HZuBAi9luuKdNTcfQ8UE/d
        jjHKZKEQRCOlLVy0Dn5ZL1RlMHsv+OzJzzvIJFTq3LQKNrzJzsNe</ds:SignatureValue>

      <ds:KeyInfo Id="KeyId-17007273">
        <wsse:SecurityTokenReference
            xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-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
```

```
        xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-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:

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 to help prevent replay attacks

The Response Structure

In response to either a Query or SOAP request, the service returns an XML data structure that conforms to an XML schema defined as part of the EC2 WSDL. The structure of an 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 `keySet` for key pairs (see the example that follows). Item elements are children of containers, and their contents vary according to the container's role.

Every EC2 response includes a request ID in a `requestId` element. The value is a unique string that AWS assigns. If you ever have issues with a particular request, AWS will ask for the request ID to help troubleshoot the issue. The following shows an example response.

```
<DescribeKeyPairsResponse xmlns="http://ec2.amazonaws.com/doc/2012-07-20/">
  <requestId>7a62c49f-347e-4fc4-9331-6e8eEXAMPLE</requestId>
  <keySet>
    <item>
      <keyName>gsg-keypair</keyName>
      <keyFingerprint>
         00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
      </keyFingerprint>
    </item>
  </keySet>
</DescribeKeyPairsResponse>
```

Available Libraries

AWS provides libraries, sample code, tutorials, and other resources for software developers who prefer to build applications using language-specific APIs instead of SOAP and Query. These libraries provide basic functions (not included in the APIs), such as request authentication, request retries, and error handling so that it is easier to get started. Libraries and resources are available for the following:

- Java

- PHP
- Python
- Ruby
- Windows and .NET

For libraries and sample code in all languages for Amazon EC2, go to
http://aws.amazon.com/code/Amazon-EC2. For libraries and sample code in all languages in Amazon
Web Services, go to Sample Code & Libraries.

Technical FAQ

Topics

This section contains answers to commonly asked questions.

General Information FAQ

How many instances can I launch?

Each AWS account has a concurrent running instance limit. For new accounts, this limit is 20. If you need more than 20 instances, please complete the Amazon EC2 Instance Request Form and your request will be considered.

How do I sign a request?

See Making API Requests (p. 532).

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.

If you need faster launch times, consider using AMIs backed by Amazon EBS.

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

Yes. For more information, see Instance Families and Types (p. 85).

Is there a REST interface to Amazon EC2?

Not at present. You can use the Query API, SOAP 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.

Instances and AMIs Information FAQ

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 back up or replicate important data.

What happens to my running instances if the machines on which they are running fail?

The instances terminate and need to be relaunched. The data on the instance stores is lost. Any data on Amazon EBS root volumes (for Amazon EBS-backed instances) is also lost by default. The default behavior of the root volume on the Amazon EBS-backed instance can be changed by setting the the value of the `deleteOnTermination` flag to `false`. For more information, see Root Device Volume (p. 117). The data on the non-root Amazon EBS volumes (if attached) is preserved, by default, on instance termination.

Always replicate important data: use Amazon EBS, or store it in Amazon S3.

What are the differences between stopping and terminating an instance?

The instance will first perform a normal shutdown in both cases. If the instance was stopped, it will then transition to a stopped state, all of its Amazon EBS volumes will remain attached, and it can then be started again at a later time. If the instance was terminated, attached Amazon EBS volumes will be deleted if the volume's `deleteOnTermination` flag is set to true. The instance itself will also be deleted, and the instance cannot be started again at a later date.

If you wish to disable termination via the `TerminateInstances` API for increased safety, ensure that the `disableApiTermination` attribute is set to true for the instance. To control the behavior of an on instance shutdown such as shutdown –h in Linux or shutdown in Windows, set the `instanceInitiatedShutdownBehavior` instance attribute to `stop` or `terminate` as desired. Instances that have Amazon EBS volume root devices will default to `stop`, and instances with instance-store root devices will always be terminated as the result of an on instance shutdown.

How does stopping and starting an instance affect how it is charged?

You will not be charged for additional instance hours while the instance is in a stopped state. A full instance hour will be charged for every transition from a stopped state to a running state, even if this happens multiple times within a single hour. If the instance type was changed while the instance was stopped, you will be charged at the new instance type's rate once the instance has been started. All of the associated Amazon EBS usage of your instance, including root device usage, will be charged at the typical Amazon EBS prices.

What can I do with a stopped instance?

You can attach or detach Amazon EBS volumes. You can use the `CreateImage` call to create an AMI from the instance. You can change the kernel, RAM disk, and instance type with the `ModifyInstanceAttribute` call. You can restart the instance with the `StartInstances` call, and terminate the instance with the `TerminateInstances` call.

What limitations exist for stopped instances?

In addition to the limit on running instances, there is an additional limit on the overall number of instances that you can have (whether running, stopped, or in any other state except for terminated.) This overall instance limit is 2 times your running instance limit. The running instance limit can be increased upon request via the instance limit request form.

Do I need to share the Amazon EBS snapshots that an AMI references in order to share the AMI?

No, only the AMI itself needs to be shared. The system will automatically provide the instance access to the referenced Amazon EBS snapshots for the purposes of the launch.

How can I control how block devices are exposed to my instance?

See Block Device Mapping (p. 488).

Operation Information FAQ

How do I handle time synchronization between instances?

You can set up NTP (Network Time Protocol). For more information, go to www.ntp.org. NTP is particularly important if you plan on using any AWS products (such as Amazon S3 or Amazon EC2) from within an instance, because 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/latest/ to retrieve various instance-specific metadata, including the instance ID. For more information, see Instance Metadata (p. 302).

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 User Data Retrieval (p. 305).

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

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.

Because 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 AWS accounts to launch your AMIs by modifying the AMI's `launchPermission` attribute. You can grant public launch permissions or explicit permissions to specific AWS accounts. For more information, see Sharing AMIs Safely (p. 62).

Why do I need to reregister a rebundled Amazon EC2 instance store-backed 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 Amazon EC2 instance store-backed AMIs after rebundling.

How can I push an Amazon S3 object to an Amazon EBS volume?

You can push an Amazon S3 object to an Amazon EBS volume by mounting the Amazon EBS volume on an Amazon EC2 instance and then using a data transfer application (e.g., FTP/SFTP, SCP) on your running instance to transfer the data.

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 and UNIX:

```
export EC2_JVM_ARGS="-Dhttp.proxyHost=http://my.proxy.com -Dhttp.proxyPort=8080"
```

Properties for configuring a proxy are described in the following table.

Setting	Description
https.proxyHost	HTTPS proxy host
https.proxyPort	HTTPS proxy port
http.proxyHost	HTTP proxy host
http.proxyPort	HTTP 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 FAQ

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 the smallest instance type available for the type of AMI you selected.

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.

How do I select the right instance type?

Amazon EC2 instances are grouped into several families. For more information, refer to Instance Families and Types (p. 85).

> **Tip**
>
> One of the advantages of Amazon EC2 is that you pay by the instance hour, which makes it convenient and inexpensive to test the performance of your application on different instance families and types. A good way to determine the most appropriate instance family and instance type is to launch test instances and benchmark your application.

When should I use High-CPU instance types (c1.medium and c1.xlarge)?

High-CPU instance types have a proportionately higher ratio of CPU to memory and are well suited for compute-intensive applications. To determine whether they are appropriate for you, launch an instance and benchmark your own application on different instance types and calculate which is most appropriate.

Which instance types are 32-bit and which are 64-bit?

All instance types can run 64-bit AMIs. The micro (t1.micro), small (m1.small), medium (m1.medium), and High-CPU medium (c1.medium) instances can also run 32-bit AMIs.

Can I launch any AMI on any type of instance?

No. Cluster compute (cc1.4xlarge and cc2.8xlarge) and cluster graphics (cg1.4xlage) instances can only run HVM AMIs. All other instances can run Windows HVM or Linux/UNIX paravirtualized AMIs.

Can I use my own kernel?

Yes. For more information, see Enabling Your Own Linux Kernels (p. 76).

Do I have to do anything special to bundle the large or extra large Amazon EC2 instance store-backed 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/UNIX 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 preinstalled 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 does I/O performance of an instance store compare to the I/O performance of an EBS volume?

The instance stores on large and extra large instances have higher and more consistent I/O performance than the original (small) instance. Amazon EBS volumes perform consistently for all instance types.

> **Note**
>
> The first write to any given block of the disk will be slower than subsequent writes. For more information, see Disk Performance Optimization (p. 486)

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 (p. 486)

IP Information FAQ

How do I host a public domain if I use DHCP to obtain an IP address?

You can use a dynamic DNS service, such as DynDNS or ZoneEdit. Alternatively, you can map an Elastic IP address to your instance and avoid the propagation delays possible with a dynamic DNS solution.

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?

The Amazon EC2 DNS servers return the internal IP address when asked about an instance's public DNS name. In this way, DNS lookups that would resolve to a public Amazon EC2 IP address will be translated to the correct internal IP address. This only works when using the Amazon EC2 DNS servers from an Amazon EC2 instance.

Why is Amazon EC2 Using NAT?

Public IP space is a limited resource. Amazon EC2 is adopting NAT to ensure 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 might 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/latest/meta-data/local-ipv4
```

To obtain the public IP address:

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

Why am I limited to 5 Elastic IP addresses?

Public (IPV4) Internet addresses are a scarce public resource. Amazon EC2 is committed to helping use that space efficiently.

By default, all accounts are limited to 5 Elastic IP addresses. If you need more than 5 Elastic IP addresses, please complete the Amazon EC2 Elastic IP Address Request Form. We will ask you to think through your use case and help us understand your need for additional addresses.

Is my Elastic IP addressed fixed to a single instance?

Unlike a traditional dedicated IP address, an Elastic IP address can be assigned to many different instances over time.

Is there a minimum usage required for Elastic IP addresses?

When operating within the 5 address limit, you can leave addresses unattached as you need. However, we reserve the right to reclaim Elastic IP addresses that are chronically underutilized.

Is there a charge for Elastic IP addresses?

You are not charged for a single Elastic IP (EIP) address on a running instance . If you associate additional EIPs with that instance, you will be charged for each additional EIP associated with that instance per hour (prorated). Additional EIPs are only available in Amazon VPC.

To ensure efficient use of Elastic IP addresses, we impose a small hourly charge when these IP addresses are not associated with a running instance, or when they are associated with a stopped instance or an unattached network interface. You can associate an Elastic IP address to an elastic network interface (ENI); however, if that ENI is not attached to an instance, you'll be charged for the Elastic IP address. To avoid charges for Elastic IP addresses that you are not using, use `ReleaseAddress`.

Do I need one Elastic IP address for each instance that I have running?

No. By default, every instance comes with a private IP address and an Internet routable public IP address. These addresses are fixed for the life of the instance (exception: Amazon EBS-backed instances typically change IP addresses when you stop and restart them). We believe the private and public IP addresses should be adequate for many applications where you do not need a long lived Internet routable end point (e.g., compute clusters, web crawling, and backend services).

What happens to an Elastic IP address if I stop the instance associated with it?

The Elastic IP address is disassociated from the instance when an EC2 instance is stopped. In Amazon VPC, when an instance is stopped, the Elastic IP address remains associated with the instance.

Why don't you use IPV6 addresses?

Because of the scarcity of IPV4 Internet address, Amazon EC2 will be actively investigating the use of IPV6 addresses. We believe this is the only tenable long term solution. We don't yet have a timeline for introducing IPV6 addresses.

Can I launch an instance with no public IP address?

You cannot currently launch an instance without a public IP address. We understand that for many applications, it is desirable to have no Internet routable IP address (e.g., internal databases).

> **Note**
>
> Instances you launch into a VPC do not have a public IP address by default. For more information about Amazon Virtual Private Cloud, go to the Amazon Virtual Private Cloud Getting Started Guide.

How long does it take to remap an Elastic IP address?

After you successfully make an API call to remap an IP address, it will usually occur within a few minutes.

Will I be charged for the time when my IP address is unattached because my instance failed?

You are not charged until your Elastic IP address has been unattached for a full hour. As long as you are monitoring your instances, you will have plenty of time to reattach your instance before the charge is metered.

Am I limited to 100 Elastic IP remaps per month?

No. The first 100 remaps per account are free. After that, there will be a charge for each remap.

Can I configure the reverse DNS record for my Elastic IP address?

Yes, you can configure the reverse DNS record of your Elastic IP address (submit the Request to Remove Email Sending Limitations form). Note that a corresponding forward DNS record pointing to that Elastic IP address must exist before we can create the reverse DNS record.

Region and Availability Zone FAQ

For a conceptual overview of regions and endpoints, see Regions and Availability Zones (p. 111).

Why aren't Regions tightly integrated with each other?

We isolate the Regions from each other to achieve greater fault tolerance, improve stability, and to help prevent issues within one Region from affecting another. To simplify using instances across regions, we provide tools such as `ec2-migrate-image` and `ec2-migrate-manifest`. For more information, go to Amazon Elastic Compute Cloud Command Line Reference.

How do I interact with EC2 in different Regions?

Use the Region-specific service endpoint for the Region you want. To get a list of Regions and their endpoints, use the `DescribeRegions` API (i.e., the **ec2-describe-regions** command), for example:

```
PROMPT> ec2-describe-regions
REGION  ap-northeast-1      ec2.ap-northeast-1.amazonaws.com
REGION  ap-southeast-1      ec2.ap-southeast-1.amazonaws.com
..
```

For more information, see Specifying the Region to Use (p. 115).

How do I launch an AMI in another Region?

For Amazon EC2 instance store-backed AMIs: Copy your AMI from its current bucket to a bucket located in the Region where you want to launch the AMI, and register the AMI. For example, to launch a US-based AMI in the EU Region, copy the AMI to an Amazon S3 bucket that was created with an EU location constraint. After the AMI is copied, you must register the AMI and use the obtained AMI ID for launches in the new Region.

Also, make sure to give read access to the bucket, image manifest, and image parts to ec2-bundled-images@amazon.com for Windows AMIs, and za-team@amazon.com for Linux AMIs.

What tools are available to help migrate my AMIs to a new Region?

The API Tools contain a new command called `ec2-migrate-image`. It is designed to help migrate AMIs to a new Region. Run ec2-migrate-image --help for more details. Also go to Amazon Elastic Compute Cloud Command Line Reference.

Can I use the same SSH key pair across Regions?

This question refers only to the key pair used for SSH connections to the instance. Don't confuse this with your AWS Account ID and other credentials, which are global and work in all Regions.

The SSH key pairs that you create with `ec2-add-keypair`, `CreateKeyPair`, or in the AWS Management Console work only in the Region where you create them. However, you can optionally create an RSA key pair with a third-party tool and upload the public key to AWS. That key pair works in all Regions. For more information, go to ec2-import-keypair in the *Amazon Elastic Compute Cloud Command Line Reference* or ImportKeyPair in the *Amazon Elastic Compute Cloud API Reference.*

How do I launch an Amazon EBS volume from a snapshot across Regions?

At this time, snapshots cannot be copied across Regions. However, data on Amazon EBS volumes can be copied across Regions out of band, For example, you can run an instance in the Region with the source volume, run an instance in the destination Region with a new volume attached, and use rsync or some other file copy mechanism to copy data.

If I make service calls to the ec2.amazonaws.com service endpoint, where will my instances launch?

They will launch in the original Amazon EC2 `ec2.us-east-1.amazonaws.com` Region.

Can instances use group-based firewall rules across Regions?

No. Group-based firewall rules only work within a Region. If you need instances to communicate with each other across Regions, you should use CIDR based firewall rules. To simplify IP address management, you can use firewall rules in combination with Elastic IP addresses.

Note

Because inter-Region traffic crosses the public Internet, encrypt all sensitive data.

How do I use the command line tools with multiple Regions?

By default, the command line tools use the original `ec2.us-east-1.amazonaws.com` endpoint (the us-east-1 Region). For information about changing the Region, see Specifying the Region to Use (p. 115).

What is the cost for data transfer between Regions?

Data transferred from one Region to another is charged at both sides at the Internet data transfer rate.

Can I assume that my Availability Zone us-east-1a is the same location as someone else's Availability Zone us-east-1a?

No. Currently, we do not support cross-account proximity. Each account's Availability Zones are independent. For example, the `us-east-1a` Availability Zone for one account might be in a different location than for another account.

How can I make sure that I am in the same Availability Zone as another developer?

We do not currently support the ability to coordinate Availability Zones between AWS accounts. We are seeking customer feedback to understand the types of use cases for proximity control between accounts. We will use this feedback to determine how and when we might provide Availability Zone control between accounts.

Regional data transfer seems like such a small charge, why are you complicating my bill with this?

We anticipate that for most common use cases, Regional data transfer will only constitute a very small portion of your monthly usage charges. There are valid use cases that involve moving large amounts of data between Availability Zones. In these cases, the Regional data transfer can be a significant cost.

We try to enable as many use cases as possible while charging you only for what you use. Because of the large potential differences in the way developers could use Regional data transfer, we think it is appropriate to break this cost out rather than amortize it across other charges.

If I have two instances in different Availability Zones, how will I be charged for Regional data transfer?

Each instance is charged for its data in and data out. Therefore, if data is transferred between these two instances, it is charged out for the first instance and in for the second instance.

If I transfer data between Availability Zones using public IP addresses, will I be charged twice for Regional data transfer (once because it crosses Availability Zones, and once because I use public IP addresses)?

No. Regional data transfer rates apply if at least one of the following cases is true, but are only charged once for a given instance even if both are true:

- The other instance is in a different Availability Zone, regardless of which type of address is used
- Public or Elastic IP addresses are used, regardless of which zone the other instance is in

Why are my Amazon EC2 resources not visible in the EU Region or other Region?

Amazon EC2 Regions are isolated from each other. Resources such as SSH key pairs from a `CreateKeyPair` call, security groups, and AMIs, are not replicated between Regions. For more information, see Resource Locations (p. 501).

Windows Instances FAQ

How can I mount or access a CD from the instance?

Select from the following:

- To create an ISO image out of the CD, upload it to your Amazon S3 bucket and download it to the instance. Then, use any standard ISO mounting tool to access it.
- To use Remote Desktop, specify the CD ROM drive letter from the **Local Resources** tab of the **Local Devices and Resources** page on the Remote Desktop client.

Monitoring, Errors, and Unexpected Behavior FAQ

How do I monitor my systems?

You can use DescribeInstances to check whether an instance appears to be running.

For more advanced monitoring, consider Amazon CloudWatch. Amazon CloudWatch runs a monitoring services that collects raw measurement data, such as CPUUtilization (percentage of Amazon EC2 compute units used by an instance) and DiskWriteBytes (number of bytes written in a minute). For more information, see Monitoring Your Instances and Volumes with CloudWatch (p. 280).

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 Security Group Concepts (p. 416)

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.

To shut down an instance, use the `TerminateInstances` call (`ec2-terminate-instances`) on the command line. You can also use `shutdown -h`, but you must verify the instance shutdown using the `DescribeInstances` call.

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 (AWS account ID) and certificate to create and upload the image. You must use the same ID and certificate to register the image with Amazon EC2.

Reserved Instances FAQs

What is a Reserved Instance?

Reserved Instances give you the option to make a low, one-time payment for each instance you want to reserve and in turn receive a significant discount on the hourly usage charge for that instance. For more information, see Reserved Instances (p. 195).

How is a Reserved Instance different than an On-Demand Instance?

Functionally, Reserved Instances and On-Demand instances are the same. They are launched and terminated in the same way, and they function identically once running. This makes it easy for you to seamlessly use both Reserved and On-Demand Instances without making any changes to your code. The only difference is that with a Reserved Instance, you pay a low, one-time payment and receive a lower usage rate to run the instance than with an On-Demand Instance.

How do I purchase and start up a Reserved Instance?

You purchase an EC2 Reserved Instance by calling the `PurchaseReservedInstancesOffering` API method. Launching a Reserved Instance is no different than launching an On-Demand Instance. You simply use the `RunInstances` call or launch an instance via the AWS Management Console. Amazon EC2 will optimally apply the cheapest rate that you are eligible for in the background.

How do I control which instances are billed at the Reserved Instance rate?

The `RunInstances` call does not distinguish between On-Demand and Reserved Instances. When computing your bill, our system automatically optimizes which instances are charged at the lower Reserved Instance rate to ensure you always pay the lowest amount.

How many Reserved Instances can I purchase?

You can purchase up to 20 Reserved Instances per Availability Zone each month with the EC2 APIs. If you need additional Reserved Instances, complete the Registration Form.

Can a Reserved Instance that I've bought for a particular instance type (i.e. High-CPU Extra Large Instance) be applied to a different instance type that I am running (i.e. Standard Large Instance)?

No. Each Reserved Instance is associated with a specific instance type, and can only be applied to that instance type for the duration of the Reserved Instance term.

Can I move a Reserved Instance from one Region or Availability Zone to another?

No. Each Reserved Instance is associated with a specific Region and Availability Zone, which is fixed for the lifetime of the Reserved Instance and cannot be changed.

Do I need to specify an Availability Zone when I launch my instances in order to take advantage of my Reserved Instances?

Yes. When you purchase a Reserved Instance you specify the Availability Zone in which you want to reserve that instance. In order to use that Reserved Instance, you need to ensure that you launch your instance in that same Availability Zone. Additionally, you can purchase a Reserved Instance in an Availability Zone where you already have a running instance, and the Reserved Instance will automatically get applied to that existing instance.

Can I cancel a Reserved Instance?

The one-time payment for a Reserved Instance is not refundable. However, you can choose not to run or entirely stop using your Reserved Instance at any time, at which point you will not incur any further usage charges.

What happens when my Reserved Instances term comes to an end?

Any instances that you have that are still running will continue to run, but will be charged at the standard On-Demand hourly rate.

When are Reserved Instances activated?

A Reserved Instance is activated once your one-time payment has successfully been authorized. You can follow the status of your Reserved Instance on the AWS Account Activity page.

How can I assign an Elastic IP address to a Reserved Instance?

You assign an Elastic IP address to a Reserved Instance in the same way that you assign an Elastic IP address to an on-demand instance. First, you acquire an Amazon EC2 Elastic IP address for your account, then you identify the instance to which you want to assign the Elastic IP address, and finally you associate it to your Reserved Instance. For more information, see .

Paid AMIs FAQ

Do you support paid AMIs that are backed by Amazon EBS?

AWS Marketplace supports AMIs backed by Amazon EBS; Amazon DevPay only supports AMIs backed by Amazon instance store.

What is the difference between AWS Marketplace and Amazon DevPay?

There are substantial differences between AWS Marketplace and Amazon DevPay. Both help customers buy software that runs on AWS, but AWS Marketplace offers a more comprehensive experience. These are the key differences for software buyers:

* AWS Marketplace offers a more Amazon.com-like shopping experience, simplifying discovery of available software.
* AWS Marketplace products work with other Amazon Web Services, such as Amazon Virtual Private Cloud (Amazon VPC) and can be run on Reserved and Spot Instances, in addition to typical On-Demand instances.
* AWS Marketplace supports EBS-backed software, whereas Amazon DevPay does not.

How can I determine the paid AMIs that are available?

You can describe the image (`ec2-describe-images`) with the `-a` flag and look for AMIs that have a product code. For example, if you run `ec2dim -a`, the result includes an AMI with the ID `ami-bd9d78d4`. This is our Demo Paid AMI with product code A79EC0DB and a product code type.

How can I determine if a particular AMI is paid?

You can describe the image (`ec2-describe-images`). An AMI is a paid AMI if a product code is returned. For example when you run `ec2dim ami-bd9d78d4`, and the results include a product code (A79EC0DB) and a product code type.

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. For example, let's say your run `ec2dim -a amazon`. When the example AMI with ID ami-bd9d78d4 is returned, it includes a product code and a product code type.

Who can use the `ec2-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/RAM disk be inherited by the rebundled AMI?

If you rebundle an AMI, it inherits the kernel and RAM disk from the source AMI unless you specify a different kernel and RAM disk.

> **Note**
>
> This is a convenience feature and not a guarantee that the kernel/RAM disk will always be attached to rebundled AMIs.

I created my paid AMIs with one AWS account, but I want to sell them using a different AWS account. Can I transfer them?

No, you can't automatically transfer AMIs from one account to another.

For AWS Marketplace, go to the AWS Marketplace Seller's Guide.

For Amazon DevPay, you would have to upload the AMIs again using the second AWS account. Then you would register them with Amazon DevPay using the second account. Alternatively, you could leave the AMIs with the original account (the AMI owner account) and register them with Amazon DevPay using another AWS account (the product owner account). You could then use the AMI owner account to associate the product code with the AMIs. However, keep in mind that only the product owner (and not the AMI owner in this case) can use the `ec2-confirm-product-instance` command, which confirms that an instance is running an AMI associated with the product owner's product code.

How do I prevent someone from stripping the product code from my paid AMI?

If you do not provide root access to your AMI, it cannot be rebundled. If you provide root access, our tools attempt to preserve the product code.

To increase security, we recommend that you configure your application to check the instance metadata to verify that the product code is intact.

Kernels and RAM Disk FAQ

What are user selectable kernels?

Amazon EC2 provides user selectable kernels which enables you to select a kernel when bundling an AMI or launching an instance. User selectable kernels are useful for keeping your instances up to date with security fixes and updates, being able to use functionality provided by new distributions, and for using specialty applications that have unique timing requirements.

How do I find user selectable kernels?

Use **ec2-describe-images -o amazon --filter "image-type=kernel"**. This lists all public kernels that are currently available.

What type of dependencies do kernels have?

Kernels are most likely to require a RAM disk that contains required drivers (e.g., Xen drivers, video drivers, and so on). If you launch a kernel without a required RAM disk, it will not work properly.

How do I know a kernel/AMI combination will work together?

If you are concerned about whether the kernel/image combination will work well together, Amazon provides several AMIs that have tested combinations that you can use as a starting point for your AMIs or AMIs that you can use as a foundation for a public AMIs. If you require a certified kernel/ AMI combination, you can find them as paid AMIs through organizations such as RedHat. For more information, see Paid AMIs (p. 31).

Can I use my own kernel?

Yes. For more information, see Enabling Your Own Linux Kernels (p. 76).

How do I know which kernels are compatible with PVGRUB?

For a list of compatible PVGRUB kernels, see Compatible PV-GRUB Kernels (p. 80). Some Linux distributions provide kernels that are not compatible with Amazon EC2. We are working with vendors to ensure that the most popular AMIs provide kernels that work with Amazon EC2, and we have tested a number of these AMIs and found them to be compatible with PVGRUB. Unfortunately, it is not possible to support every kernel that is or can be compiled. To avoid the situation in which a kernel does not work consistently or at all, we recommend that you use a known good kernel, select a non-PVGRUB AKI, or seek support from your AMI vendor.

Error Messages FAQ

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 AWS accounts, the limit is 20.

If you need additional capacity, please complete the form at http://aws.amazon.com/contact-us/ec2-request.

Why can't I retrieve my instance-specific data from within a running instance when querying http://169.254.169.254/latest/?

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 you still experience problems retrieving the data after relaunching your instance, check the following:

- Verify you are using the correct base URI (http://169.254.169.254/latest/)
- Verify you are using the correct URI for the data you are trying to retrieve. Depending on the data, a trailing '/' might be required
- Verify you specified 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 time zone. For more information, go to NTP.

My instance is in a "running" state but shows 272 for its state code. What does that mean?

This typically indicates some sort of issue with the host running the instance. First try rebooting the instance (through the AWS Management Console, or with `ec2-reboot-instances` or `RebootInstances`). Caution: for Windows instances, this operation performs a hard reboot that might result in data corruption.

If the reboot doesn't work, post a message to the EC2 forums with the instance ID. Typically someone from the EC2 team can get your instance back to a normal state. You can search for "code 272" in the forums and find previous messages others have posted.

Miscellaneous FAQ

What runlevel do instances start in?
If you use one of the Xen provided kernels to boot your EC2 instance it will default to run level 4. However if you use PV-GRUB to boot your own kernel inside of the instance, the instance will then default to the internally configured run level.

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.

Amazon EC2 Resources

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

Resource	Description
Amazon Elastic Compute Cloud Getting Started Guide	Provides a quick tutorial of the service based on a simple use case. Examples and instructions are included.
Amazon Elastic Compute Cloud User Guide	Provides conceptual information about Amazon EC2 and describes how to use Amazon EC2 features using the AWS Management Console, command line tools, and Query API.
Amazon Elastic Compute Cloud API Reference	Contains a comprehensive description of the API actions, data types, and errors.
Amazon Elastic Compute Cloud Command Line Reference	Contains a comprehensive description of all the command line tools and their options.
Amazon EC2 Technical FAQ	Covers the top questions developers have asked about this product.
Amazon EC2 Release Notes	Give a high-level overview of the current release. They specifically note any new features, corrections, and known issues.
AWS Developer Resource Center	A central starting point to find documentation, code samples, release notes, and other information to help you build innovative applications with AWS.
AWS Management Console	The console lets you perform most of the functions of Amazon EC2 and other AWS products without programming.
Discussion Forums	A community-based forum for developers to discuss technical questions related to Amazon Web Services.
AWS Support Center	The home page for AWS Technical Support, including access to our Developer Forums, Technical FAQs, Service Status page, and AWS Premium Support (if you are subscribed to this program).

Resource	Description
AWS Premium Support Information	The primary web page for information about AWS Premium Support, a one-on-one, fast-response support channel to help you build and run applications on AWS Infrastructure Services.
Amazon EC2 Product Information	The primary web page for information about Amazon EC2.
Form for questions related to your AWS account: Contact Us	This form is *only* for account questions. For technical questions, use the Discussion Forums.
Terms of Use	Detailed information about the copyright and trademark usage at Amazon.com and other topics.

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.
Amazon EBS	A type of storage that enables you to create volumes that can be mounted as devices by Amazon EC2 instances. Amazon EBS volumes behave like raw unformatted external block devices. They have user supplied device names and provide a block device interface. You can load a file system on top of Amazon EBS volumes, or use them just as you would use a block device.
Amazon EBS-backed AMI	An instance launched from an AMI backed by Amazon EBS uses an Amazon EBS volume as its root device. See Amazon EBS.
Instance store-backed AMI	An instance launched from an Amazon S3backed AMI uses an instance store as its root device. See instance store.
Availability Zone	A distinct location within a Region that is engineered to be insulated from failures in other Availability Zones and provides inexpensive, low latency network connectivity to other Availability Zones in the same Region.
compute unit	An Amazon-generated measure that enables you to evaluate the CPU capacity of different Amazon EC2 instance types.
EBS	See Amazon EBS.
Elastic Block Store	See Amazon EBS.
elastic IP address	A static public IP address designed for dynamic cloud computing. Elastic IP addresses are associated with your account, not specific instances. Any elastic IP addresses that you associate with your account remain associated with your account until you explicitly release them. Unlike traditional static IP addresses, however, elastic IP addresses allow you to mask instance or Availability Zone failures by rapidly remapping your public IP addresses to any instance in your account.
ephemeral store	See *instance store*.
explicit launch permission	Launch permission granted to a specific AWS account.
filter	Criterion you specify to limit the results when you list or describe your EC2 resources.

gibibyte (GiB)	A contraction of giga binary byte, a gibibyte is 2^30 bytes or 1,073,741,824 bytes. A gigabyte is 10^9 or 1,000,000,000 bytes.
group	See security group.
image	See *Amazon machine image*.
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.
instance store	Every instance includes a fixed amount of storage space on which you can store data. This is not designed to be a permanent storage solution. If you need a permanent storage system, use Amazon EBS.
instance type	A specification that defines the memory, CPU, storage capacity, and hourly cost for an instance. Some instance types are designed for standard applications, whereas others are designed for CPU-intensive applications, or memory-intensive applications, etc.
launch permission	AMI attribute allowing AWS accounts to launch an AMI
Linux	Amazon EC2 instances are available for many operating platforms, including Linux, Solaris, Windows, and others.
maximum price	The maximum price you will pay to launch one or more Spot Instances. If your maximum price exceeds the Spot Price and your restrictions are met, Amazon EC2 launches instances on your behalf.
paid AMI	An AMI that you sell to other Amazon EC2 users. For more information, refer to the *Amazon DevPay Developer Guide*.
private IP address	All Amazon EC2 instances are assigned two IP addresses at launch: a private address (RFC 1918) and a public address that are directly mapped to each other through Network Address Translation (NAT).
public AMI	An AMI that all AWS accounts have launch permissions for.
public data sets	Sets of large public data sets that can be seamlessly integrated into AWS cloud-based applications. Amazon stores the data sets at no charge to the community and, like with all AWS services, you pay only for the compute and storage you use for their your applications. These data sets currently include data from the Human Genome Project, the U.S. Census, Wikipedia, and other sources.
public IP address	All Amazon EC2 instances are assigned two IP addresses at launch: a private address (RFC 1918) and a public address that are directly mapped to each other through Network Address Translation (NAT).
region	A geographical area in which you can launch instances (e.g., US, EU).
reservation	A collection of instances started as part of the same launch request.
Reserved Instance	An additional Amazon EC2 pricing option. With Reserved Instances, you can make a low one-time payment for each instance to reserve and receive a significant discount on the hourly usage charge for that instance.
resource	A general term that refers to the objects you work with in Amazon EC2. This includes instances, images, Amazon EBS volumes, snapshots, etc.

security group	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.
shared AMI	AMIs that developers build and make available for other AWS developers to use.
Solaris	Amazon EC2 instances are available for many operating platforms, including Linux, Solaris, Windows, and others.
snapshot	Amazon EBS provides the ability to create snapshots or backups of your Amazon EBS volumes and store them in Amazon S3. You can use these snapshots as the starting point for new Amazon EBS volumes and to protect your data for long term durability.
Spot Instance	A type of instance that you can bid on to take advantage of unused Amazon EC2 capacity.
Spot Price	The current price for Spot Instances. If your Spot Instance request exceeds this price and your restrictions are met, Amazon EC2 launches instances on your behalf.
supported AMIs	These AMIs are similar to paid AMIs, except that you charge for software or a service that customers use with their own AMIs.
tag	Metadata of your choice (consisting of up to 10 key-value pairs) that you can optionally assign to EC2 resources.
tebibyte (TiB)	A contraction of tera binary byte, a tebibyte is 2^40 bytes or 1,099,511,627,776 bytes. A terabyte is 10^12 or 1,000,000,000,000 bytes.
UNIX	Amazon EC2 instances are available for many operating platforms, including Linux, Solaris, Windows, and others.
Windows	Amazon EC2 instances are available for many operating platforms, including Linux, Solaris, Windows, and others.

Document History

The following table describes important additions to the Amazon EC2 documentation set. We also update the documentation frequently to address the feedback that you send us.

Current API version: 2012-07-20.

Feature	API Version	Description	Release Date
Provisioned IOPS (input/output operations per second) for Amazon EBS	2012-07-20	Provisioned IOPS volumes deliver predictable, high performance for I/O intensive workloads, such as database applications, that rely on consistent and fast response times. For more information, see EBS Volume Types (p. 436).	31 July 2012
High I/O instances for Amazon EC2	2012-06-15	High I/O instances provides very high, low latency, disk I/O performance using SSD-based local instance storage. For more information, see High I/O Instances (p. 97).	13 July 2012
IAM roles on Amazon EC2 instances	2012-06-01	IAM roles for Amazon EC2 provide: • AWS access keys for applications running on Amazon EC2 instances. • Automatic rotation of the AWS access keys on the Amazon EC2 instance. • Granular permissions for applications running on Amazon EC2 instances that make requests to your AWS services.	11 June 2012

Feature	API Version	Description	Release Date
Spot Instance features that make it easier to get started and handle the potential of interruption.		Using Auto Scaling, you can now manage your Spot Instances: • Place bids for Amazon EC2 Spot Instances using Auto Scaling launch configurations, and set up a schedule for placing bids for Spot Instances. For more information, see Managing Spot Instances with Auto Scaling (p. 144). • Get notifications when instances are launched or terminated. For more information, see Get Notifications through Auto Scaling for Spot Instances (p. 160). • Use AWS CloudFormation templates to launch Spot Instances in a stack with AWS resources. For more information, see Using CloudFormation Templates to Launch Spot Instances (p. 156).	7 June 2012
EC2 instance export and timestamps for status checks for Amazon EC2	2012-05-01	Added support for exporting Windows Server instances that you originally imported into EC2. Added support for timestamps on instance status and system status to indicate the date and time that a status check failed.	25 May 2012
EC2 instance export, and timestamps in instance and system status checks for Amazon VPC	2012-05-01	Added support for EC2 instance export to Citrix Xen, Microsoft Hyper-V, and VMware vSphere. Added support for timestamps in instance and system status checks.	25 May 2012
Cluster Compute Eight Extra Large instances	2012-04-01	Added support for cc2.8xlarge instances in Amazon Virtual Private Cloud (Amazon VPC).	26 April 2012
AWS Marketplace AMIs	2012-04-01	Added support for AWS Marketplace AMIs.	19 April 2012
New Linux AMI release		Added information about the release of Amazon Linux AMI 2012.03. For more information, see Amazon Linux AMIs (p. 70).	28 March 2012
New AKI version		Added information about the release of new AKI version 1.03 and the release of AKIs for the AWS GovCloud (US) region. For more information, see Enabling Your Own Linux Kernels.	28 March 2012
Medium instances, support for 64-bit on all AMIs, and a Java-based SSH Client		Added support for a new instance type and 64-bit information. Added procedures for using the Java-based SSH client to connect to Linux/UNIX instances.	27 March 2012
Reserved Instance Pricing Tiers		Added a new section discussing how to take advantage of the discount pricing that is built into the Reserved Instance Pricing Tiers. For more information, see Understanding Reserved Instance Pricing Tiers (p. 203).	5 March 2012

Feature	API Version	Description	Release Date
Elastic Network Interfaces (ENIs) for EC2 instances in Amazon Virtual Private Cloud		Added new section about elastic network interfaces (ENIs) for EC2 instances in a VPC. For more information, see Elastic Network Interfaces (p. 365).	1 2 reD 1102
New GRU Region and AKIs		Added information about the release of new AKIs for the SA-East-1 Region. This release deprecates the AKI version 1.01. AKI version 1.02 will continue to be backward compatible.	4 1 reD 1102
New offering types for Amazon EC2 Reserved Instances	2011-11-01	You can choose from a variety of Reserved Instance offerings that address your projected use of the instance: *Heavy Utilization*, *Medium Utilization*, and *Light Utilization*. See Reserved Instances (p. 195).	1 0 reD 1102
Amazon EC2 instance status		You can view additional details about the status of your instances, including scheduled events planned by AWS that might have an impact on your instances. These operational activities include instance reboots required to apply software updates or security patches, or instance retirements required where there are hardware issues. See Monitoring the Status of Your Instances (p. 291).	6 1 reN 1102
Amazon EC2 Cluster Compute Instance Type		Added support for Cluster Compute Eight Extra Large (cc2.8xlarge) to Amazon EC2.	4 1 reN 1102
New PDX Region and AKIs		Added information about the release of new AKI's for the new US-West 2 Region.	8 reN 1102
Amazon EC2 Spot Instances in Amazon VPC		Added information about the support for Amazon EC2 Spot Instances in Amazon VPC. With this update, users will be able to launch Spot Instances in the Amazon Virtual Private Cloud (Amazon VPC). By launching Spot Instances in Amazon VPC, users of Spot Instances can enjoy all of the controls and advanced security options of Amazon VPC. For more information, see Launch Spot Instances in Amazon Virtual Private Cloud (p. 158).	1 reO 1102
New Linux AMI release		Added information about the release of Amazon Linux AMI 2011.09. This update removes the beta tag from the Amazon Linux AMI, supports the ability to lock the repositories to a specific version, and provides for notification when updates are available to installed packages including security updates. For more information, see Amazon Linux AMIs (p. 70).	6 2 reS 1102

Feature	API Version	Description	Release Date
Simplified VM import process for users of the CLI tools		The VM Import process for CLI users is simplified with the enhanced functionality of `ec2-import-instance` and `ec2-import-volume`, which now will perform the upload of the images into Amazon EC2 after creating the import task. In addition, with the introduction of the `ec2-resume-import` command, users can restart an incomplete upload at the point the task stopped. For more information, see Importing Your Virtual Machine into Amazon EC2 (p. 247) in the *Amazon Elastic Compute Cloud User Guide*.	15 September 2011
Support for importing in VHD file format		VM Import can now import virtual machine image files in VHD format. The VHD file format is compatible with the Citrix Xen and Microsoft Hyper-V virtualization platforms. With this release, VM Import now supports RAW, VHD and VMDK (VMware ESX-compatible) image formats. For more information, see Using the Command Line Tools to Import Your Virtual Machine to Amazon EC2 (p. 240) in the *Amazon Elastic Compute Cloud User Guide*.	24 August 2011
Support for Microsoft Windows Server 2003 R2		VM Import now supports Windows Server 2003 (R2). With this release, VM Import supports all versions of Microsoft Windows Server supported by Amazon EC2.	24 August 2011
Update to the Amazon EC2 VM Import Connector for VMware vCenter		Added information about the 1.1 version of the Amazon EC2 VM Import Connector for VMware vCenter virtual appliance (Connector). This update includes proxy support for Internet access, better error handling, improved task progress bar accuracy, and several bug fixes. For more information, see Using Instances of Your Virtual Machine in Amazon EC2 (p. 224).	27 June 2011
Enabling Linux AMI to run user-provided kernels		Added information about the AKI version change from 1.00 to 1.02. This version updates the PVGRUB to address launch failures associated with t1.micro Linux instances. For more information, go to Enabling Your Own Linux Kernels.	20 June 2011
Spot Instances Availability Zone pricing changes		Added information about the Spot Instances Availability Zone pricing feature. In this release, we've added new Availability Zone pricing options as part of the information returned when you query for Spot Instance requests and Spot Price history. These additions make it easier to determine the price required to launch a Spot Instance into a particular Availability Zone. For more information, see Spot Instances (p. 125).	26 May 2011
AWS Identity and Access Management		Added information about AWS Identity and Access Management (IAM), which enables users to specify which Amazon EC2 actions a user can use with Amazon EC2 resources in general. For more information, see AWS Identity and Access Management (p. 356).	26 April 2011

Feature	API Version	Description	Release Date
Enabling Linux AMI to run user-provided kernels		Added information about enabling a Linux AMI to use PVGRUB Amazon Kernel Image (AKI) to run a user-provided kernel. For more information, go to Enabling Your Own Linux Kernels.	26 April 2011
Dedicated instances		Launched within your Amazon Virtual Private Cloud (Amazon VPC), Dedicated Instances are instances that are physically isolated at the host hardware level. Dedicated Instances let you take advantage of Amazon VPC and the AWS cloud, with benefits including on-demand elastic provisioning and pay only for what you use, while isolating your Amazon EC2 compute instances at the hardware level. For more information, go to Using EC2 Dedicated Instances Within Your VPC in the *Amazon Virtual Private Cloud User Guide*.	27 March 2011
Reserved Instances updates to the AWS Management Console		Updates to the AWS Management Console make it easier for users to view their Reserved Instances and purchase additional Reserved Instances, including Dedicated Reserved Instances. For more information, see Reserved Instances (p. 195).	27 March 2011
Support for Windows Server 2008 R2		Amazon EC2 now provides you with several pre-configured Windows Server 2008 R2 AMIs. These AMIs are immediately available for use in every Region and in most 64-bit instance types, excluding t1.micro and HPC families. The AMIs will support multiple languages. For more information, see Windows Instance Types (p. 89).	15 March 2011
New Amazon Linux reference AMI		Added information about the new Amazon Linux reference AMI, which replaces the CentOS reference AMI. Removed information about the CentOS reference AMI, including the section named Correcting Clock Drift for Cluster Instances on CentOS 5.4 AMI. For more information, see Reference AMIs (p. 100).	15 March 2011
Metadata information		Added information about metadata to reflect changes in the 2011-01-01 release. For more information, see Instance Metadata (p. 302) and Metadata Categories (p. 308).	14 March 2011
Amazon EC2 VM Import Connector for VMware vCenter		Added information about the Amazon EC2 VM Import Connector for VMware vCenter virtual appliance (Connector). The Connector is a plug-in for VMware vCenter that integrates with VMware vSphere Client and provides a graphical user interface that you can use to import your VMware virtual machines to Amazon EC2. For more information, see Using Instances of Your Virtual Machine in Amazon EC2 (p. 224).	3 March 2011
Force volume detachment		You can now use the AWS Management Console to force the detachment of an Amazon EBS volume from an instance. For more information, see Detaching an Amazon EBS Volume from an Instance (p. 471).	23 February 2011

Feature	API Version	Description	Release Date
Instance termination protection		You can now use the AWS Management Console to prevent an instance from being terminated. For more information, you can see Enabling Termination Protection for an Instance (p. 315).	8 February 2011
Correcting Clock Drift for Cluster Instances on CentOS 5.4 AMI		Added information about how to correct clock drift for cluster instances running on Amazon's CentOS 5.4 AMI.	5 January 2011
Restructured User Guide		Implemented the following updates to the Amazon EC2 User Guide: • Consolidated all developer guide information into this user guide. • Restructured and updated the information about creating AMIs. For more information, see Creating Your Own AMIs (p. 35). • Updated the introduction to the user guide. For more information, see What is Amazon EC2? (p. 3)	4 January 2011
VM Import		Added information about VM Import, which allows you to import a virtual machine or volume into Amazon EC2. For more information, see Using the Command Line Tools to Import Your Virtual Machine to Amazon EC2 (p. 240).	15 December 2010
Basic monitoring for instances		Added information about basic monitoring for EC2 instances. For more information, see Monitoring Instances (p. 280).	12 December 2010
Cluster GPU instances		Amazon EC2 offers cluster GPU instances (cg1.4xlarge) for high-performance computing (HPC) applications. For more information, see Overview (p. 99).	14 November 2010
Filters and Tags		Added information about listing, filtering, and tagging resources. For more information, see Listing and Filtering Your Resources (p. 503) and Tagging Your Resources (p. 508).	19 September 2010
Idempotent Instance Launch		Added information about ensuring idempotency when running instances. For more information, see Ensuring Idempotency (p. 300).	19 September 2010
Micro instances		Amazon EC2 offers the t1.micro instance type for certain types of applications. For more information, see Micro Instances (p. 90).	8 September 2010
AWS Identity and Access Management for Amazon EC2		Amazon EC2 now integrates with AWS Identity and Access Management (IAM). For more information, see AWS Identity and Access Management (p. 9).	2 September 2010
Cluster instances		Amazon EC2 offers cluster compute instances for high-performance computing (HPC) applications. For more information, see Overview (p. 99).	12 July 2010

Feature	API Version	Description	Release Date	
Amazon VPC IP Address Designation		Amazon VPC users can now specify the IP address to assign an instance launched in a VPC. For more information, see IP Addresses for Instances in a VPC (p. 384).	2 yluJ 0102	1
Amazon CloudWatch Monitoring for Amazon EBS Volumes		Amazon CloudWatch monitoring is now automatically available for Amazon EBS volumes. For more information, see Monitoring Amazon EBS Volumes (p. 289).	4 enuJ 0102	1
High-memory extra large instances		Amazon EC2 now supports a High-Memory Extra Large (m2.xlarge) instance type. For more information, see Available Instance Types (p. 86).	2 yaruF 0102	2
Reserved Instances with Windows		Amazon EC2 now supports Reserved Instances with Windows. For more information about Reserved Instances, see Reserved Instances (p. 195).	2 yaruF 0102	2

Index