Eucalyptus 3.3.2 User Guide

2013-09-27 Eucalyptus Systems

Contents

Welcome

Eucalyptus is open source software for building AWS-compatible private and hybrid clouds. As an Infrastructure as a Service product, Eucalyptus allows you to flexibly provision your own collections of resources (both compute and storage), on an as-needed basis. It also features high-availability configurations to ensure the robustness of your cloud.

Whether you're looking for an on-premise complement to your public cloud setup, or already have a virtualized datacenter and now want to move into a private or hybrid cloud setup, Eucalyptus can help you get started today. Download Faststart and have your cloud up and running in half an hour, or take advantage of Eucalyptus' seamless interoperation with Amazon Web Services' EC2 and S3 public cloud services, for an enterprise-grade hybrid cloud platform out of the box.

Eucalyptus Features

Eucalyptus offers ways to implement, manage, and maintain your own collection of virtual resources (machines, network, and storage). The following is an overview of these features.

AWS API compatibility	Eucalyptus provides API-compatibility with Amazon Web Services, to allow you to use familiar tools and commands to provision your cloud.
High-Availability configurations	Eucalyptus offers a High-Availability configuration, to ensure that your cloud is robust and fault-tolerant.
Block- and bucket-based storage abstractions	Eucalyptus provides storage options compatible with Amazon's EBS (block-based) and S3 (bucket-based) storage products.
Self-service capabilities	Eucalyptus offers a User Console, allowing your users to request the resources they need, and automatically provisioning those resources where available.
Graphical cloud-management console	The Eucalyptus Dashboard provides cloud administrators with a powerful graphical interface for cloud management, configuration, provisioning and reporting.

Resource Management

Eucalyptus offers tools to seamlessly manage a variety of virtual resources. The following is an overview of the types of resources your cloud platform.

SSH Key Management	Eucalyptus employs public and private keypairs to validate your identity when you log into VMs using SSH. You can add, describe, and delete keypairs.
Image Management	Before running instances, someone must prepare VM images for use in the cloud. This can be an administrator or a user. Eucalyptus allows you to bundle, upload, register, describe, download, unbundle, and deregister VM images.
Linux Guest OS Support	Eucalyptus lets you run your own VMs in the cloud. You can run, describe, terminate, and reboot a wide variety of Linux-based VMs that were prepared using image management commands.
Windows Guest OS Support	Eucalyptus allows you to run, describe, terminate, reboot, and bundle instances of Windows VMs.
IP Address Management	Eucalyptus can allocate, associate, disassociate, describe, and release IP addresses. Depending on the networking mode, you might have access to public IP addresses that are not statically associated with a VM (elastic IPs). Eucalyptus provides tools to allow users to reserve and dynamically associate these elastic IPs with VMs.
Security Group Management	Security groups are sets of firewall rules applied to VMs associated with the group. Eucalyptus lets you create, describe, delete, authorize, and revoke security groups.

Volume and Snapshot Management	Eucalyptus allows you to create dynamic block volumes. A dynamic block volume is similar to a raw block storage device that can be used with VMs. You can create, attach, detach, describe, bundle, and delete volumes. You can also create and delete snapshots of volumes and create new volumes from snapshots.

Who Should Read this Guide?

This guide is for Eucalyptus users who wish to run and manage Linux-based and Windows-based virtual machines (VMs) within a Eucalyptus cloud.

What's in this Guide?

This guide contains instructions for users of the Eucalyptus cloud platform. While these instructions apply generally to all client tools capable of interacting with Eucalyptus, the primary focus is on the use of Euca2ools (Eucalyptus command line tools). The following is an overview of the contents of this guide.

Guide version: Build 1873 (2013-09-27 09:59:39)

Getting Started

This section helps you get started using your Eucalyptus cloud. First, we present an overview of the features available to end-users. Then we show you how to sign up for an account, obtain credentials, and set the environment variables that enable you to interact with Eucalyptus via client tools, such as Euca2ools—Eucalyptus command-line tools. Note that Eucalyptus is compatible with Amazon EC2, thus EC2 users can continue using ec-2 tools with Eucalyptus.

In order to use Eucalyptus, you need to:

1. *Sign Up for an Account*
2. *Get User Credentials*
3. *Set Up Euca2ools*

After you have the credentials and a way to access Eucalyptus, you can start using your cloud.

Sign Up for an Account

Before you can access your Eucalyptus cloud, you must first sign up for an account using the Eucalyptus Administrator Console.

Important: You won't be able to use Eucalyptus until your administrator has approved and enabled your account. The more complete the information you provide on your account application the easier for the administrator to verify your identity.

To sign up for an account:

1. Open your browser to the Eucalyptus Administrator Console at https://<your_CLC_IP_address>:8443/.
   Ask your system administrator for the URL if you don't know it.
2. Click **Apply** to access the application form.
3. Fill out the Eucalyptus account application form.
   An approval email is sent to your mailbox.
4. Click the URL in the confirmation email that you received from the cloud administrator.

This creates an account with an admin user. You can now log in to the Eucalyptus Administrator Console as the admin user, using the username and password that you chose when filling out the application form.

Get User Credentials

You must have proper credentials to use client tools (such as Euca2ools) to interact with the Eucalyptus cloud. After you have signed up for an account and have received approval from the administrator, you can log in to the Eucalyptus Administrator Console as the admin user and obtain these credentials.

To get user credentials

1. Log in to the Eucalyptus Administrator Console using your username and password.
2. Click your username at the top of the screen and then click **Download new credentials**.
   The download contains a zip-file with your public/private key pair, a bash script, and several other required files.
3. Unzip your credentials zip file to a directory of your choice. In the following example we download the credentials zip file to ~/.euca, then change access permissions, as shown:

```
mkdir ~/.euca
cd ~/.euca
unzip <filepath>/euca2-<user>-x509.zip
```

```
chmod 0700 ~/.euca
chmod 0600 *
```

You're now ready to use the command-line interface (CLI), called "Euca2ools."

Set Up Euca2ools

Eucalyptus provides a command-line interface (CLI), called Euca2ools. The remainder of this guide uses Euca2ools. These tools conform to the command-line tools that Amazon distributes as part of their EC2 service (EC2 tools).

Tip: You can use either help pages or man pages to view information about Euca2ools commands and associated options. To see a help page, enter `<commandName> --help`. To see a man page, enter `man <commandName>`. For example, to get help for the `euca-bundle-images` command, enter either `euca-bundle-image --help` or `man euca-bundle-image`.

Overview of Euca2ools

Euca2ools is the Eucalyptus command line interface for interacting with web services. This set of tools was written in Python.

Most Euca2ools commands are compatible with Amazon's EC2, S3, IAM, Auto Scaling, Elastic Load Balancing, and CloudWatch services and generally accept the same options and honor the same environment variables. This means that you can use Euca2ools with both the Eucalyptus cloud-computing platform and Amazon EC2, S3, IAM, Auto Scaling, Elastic Load Balancing, and CloudWatch. A few commands are specific to Eucalyptus and are noted in the command description.

Note: Some operations in Euca2ools are not supported in certain Eucalyptus networking modes. For example, security groups and elastic IPs are not supported in both Static and System networking modes. See your cloud administrator for details. For more information about networking modes, see the *Eucalyptus Installation Guide*.

Installing Euca2ools

This section covers how to install Euca2ools.

- *Installing Euca2ools on RHEL / CentOS*
- *Installing Euca2ools on Mac OS X*

Setting Environment Variables

Euca2ools uses two kinds of credentials to authenticate user identity: X.509 PEM-encoded certificates and access keys. In addition, you must also specify service endpoints.

You must either define a set of environment variables in advance or use command-line options to allow Euca2ools to communicate with the cloud and verify user identity.

You can specify each value individually using the command line, set all of them collectively by sourcing the eucarc file (for more information, see *Sourcing a Eucarc File*), or use a Euca2ools configuration file (for more information, see *Working with Euca2ools Configuration Files*).

Variable	Option	Description
EC2_URL	`-U, --url <url>`	http://<host_ip>:8773/services/Eucalyptus
S3_URL	`-U, --url <url>`	http://<host_ip>:8773/services/Walrus
EUCALYPTUS_CERT	`--ec2cert_path <file>`	Path to cloud certificate
EC2_CERT	`-c, --cert <file>`	Path to your PEM-encoded certificate

Variable	Option	Description
EC2_PRIVATE_KEY	`-k, -privatekey <file>`	Path to your PEM-encoded private key
EC2_ACCESS_KEY	`-a, --access-key <key>`	Your access key ID
EC2_SECRET_KEY	`-s, --secret-key <key>`	Your secret access key

Sourcing a Eucarc File

`eucarc` is a resource configuration file that defines the correct values for each associated variable. `eucarc` is in your credentials zip file. Sourcing eucarc sets these variables to the correct values.

Note: When you source eucarc, you are only setting the correct values for the session. If you open a new terminal you must source eucarc again.

To source the eucarc file, enter the following:

Open a terminal and enter the following:

```
source eucarc
```

The appropriate environment variables are now set and you are ready to use Euca2ools to interact with your Eucalyptus cloud.

Using Images

An image is a snapshot of a system's root file system and it provides the basis for instances. When you run a new virtual machine, you choose a machine image to use as a template. The new virtual machine is then an instance of that machine image that contains its own copy of everything in the image. The instance keeps running until you stop or terminate it, or until it fails. If an instance fails, you can launch a new one from the same image. You can create multiple instances of a single machine image. Each instance will be independent of the others.

You can use a single image or multiple images, depending on your needs. From a single image, you can launch different types of instances. An *instance type* defines what hardware the instance has, including the amount of memory, disk space, and CPU power.

A machine image contains all the information needed to boot instances of your software. For example, a machine image might contain software to act as an application server or Hadoop node.

Note: This section describes instances that are not backed by dynamic block volumes, or Eucalyptus block storage (EBS) devices. For information about EBS, see *Using EBS*.

Available Images

Existing machine images available at the *Eucalyptus Machine Images* page.

You might find that an existing machine image meets your needs. However, you might want to customize a machine image or create a machine image for your own use. For more information about creating your own machine image, see *Creating an Image*.

Image Types

Machine images are either backed by persistent storage or backed by instance store. An image backed by persistent storage means that the root device is a snapshot and appears as a persistent volume when an instance is launched from the machine image. A machine that is backed by instance store means that the root device is stored in Walrus and appears as instance store when an instance is launched from the machine image

.

Image Overview

An image defines what will run on a guest instance with your Eucalyptus cloud. Typically, an image contains one of the Linux distributions like CentOS, Fedora, Ubuntu, Debian or others. It could also contain one of the supported Windows server versions. The format for these is identical.

Normally when we use the term "image" we mean the root file system. Once bundled, uploaded, and registered with Eucalyptus, such an image is known as a Eucalyptus machine image (EMI).

There are, however, other types of images that support the EMI. They are the kernel (EKI) and ramdisk (ERI). They contain kernel modules necessary for proper functioning of the image. Often, one set of these ERIs and EKIs are used by multiple EMIs. Once loaded into the Eucalyptus cloud, the EKI and ERI are referred to by the image and you don't have much interaction with them directly.

Tip: When you run an image, you can override the image's associated kernel and ramdisk if necessary (for example, if you want to try out another kernel. For more information, see *Associate a Kernel and Ramdisk*.

To help get you started, Eucalyptus provides pre-packaged virtual machines that are ready to run in your cloud. You can get them at the *Eucalyptus Machine Images* page. Each Eucalyptus image from this site comes bundled with a correspoding EKI and ERI. You can manually download these images from the web page, or you can use the Eualyptus Image Store commands to list and describe these images, as well as to install an image in your cloud. For more information see the Eucalyptus Image Store section in the *Euca2ools Reference Guide*.

If you find that the pre-packaged images don't meet your needs, you can add an image from an existing, non-registered image, or create your own image.

Once you've selected and downloaded the image(s) you plan to use, read the details in the following section for directions about how to bundle, upload and register the images with your Eucalyptus cloud.

To view the status of your newly created and registered EKI, ERI, and EMI images, use the `euca-describe-images` command. You can also view image status using the Eucalyptus Administrator Console's **Images** page.

Once you have added your image to Eucalyptus, see the *User Guide* for information about launching and using instances based on the image.

Image Tasks

This section explains the tasks that you can perform on images in Eucalyptus.

You can perform the following image-related tasks listed in the following sections:

- *Add an Image*
- *Browse and Install Images from EuStore*
- *Associate a Kernel and Ramdisk*
- *Modify an Image*
- *Creating an Image*
- *Bundle an Image for Amazon EC2*
- *Bundle a Windows Instance*

Add an Image

An image is the basis for instances that you spin up for your computing needs. If you have access to images that meet your needs, you can skip this section. This section explains how to add an image to your Eucalyptus cloud that is customized for your needs.

There are a few ways you can add an image to Eucalyptus:

- **Add an image based on an existing image.** If you have an image already, read the rest of this section. Eucalyptus has stock images available to help you get started right away. You can find links to these images in the Eucalyptus Administrator Console's start page. You can also get images from the *Eucalyptus Machine Images* page.
- Add an image that you modify from an existing image. If you have an image that you want to modify, see *Modify an Image*.
- Add an image that you create. If you want to create a new image, see *Creating an Image*.

Each of these three ways has a different first step, but the way you get image to Eucalyptus is the same. To enable an image as an executable entity, you do the following:

1. Bundle a root disk image and kernel/ramdisk pair
2. Upload the bundled image to Walrus bucket storage
3. Register the data with Eucalyptus

Important: Note that while all users can bundle, upload and register images, only users under the `administrator` account have the required permissions to upload and register kernels and ramdisks.

Once you have an image that meets your needs, perform the tasks listed in this section to add the image to your cloud.

Add a Kernel

When you add a kernel to Walrus, you bundle the kernel file, upload the file to a bucket in Walrus that you name, and then register the kernel with Eucalyptus.

To add a kernel to Walrus:

Use the following three commands:

```
euca-bundle-image -i <kernel_file> --kernel true
euca-upload-bundle -b <kernel_bucket> -m /tmp/<kernel_file>.manifest.xml
euca-register <kernel_bucket>/<kernel_file>.manifest.xml
```

For example:

```
euca-bundle-image -i
euca-fedora-10-x86_64/xen-kernel/vmlinuz-2.6.27.21-0.1-xen --kernel
 true
...
Generating manifest /tmp/vmlinuz-2.6.27.21-0.1-xen.manifest.xml

euca-upload-bundle -b example_kernel_bucket -m
/tmp/vmlinuz-2.6.27.21-0.1-xen.manifest.xml
...
Uploaded image as
example_kernel_bucket/vmlinuz-2.6.27.21-0.1-xen.manifest.xml

euca-register
example_kernel_bucket/vmlinuz-2.6.27.21-0.1-xen.manifest.xml
IMAGE eki-XXXXXXXX
```

Where the returned value `eki-XXXXXXXX` is the unique ID of the registered kernel image.

Add a Ramdisk

When you add a ramdisk to Walrus, you bundle the ramdisk file, upload the file to a bucket in Walrus that you name, and then register the ramdisk with Eucalyptus.

To add a ramdisk to Walrus:

Use the following three commands:

```
euca-bundle-image -i <ramdisk_file> --ramdisk true
euca-upload-bundle -b <ramdisk_bucket> -m /tmp/<ramdisk_file>.manifest.xml
euca-register <ramdisk_bucket>/<ramdisk_file>.manifest.xml
```

For example:

```
euca-bundle-image -i
euca-fedora-10-x86_64/xen-kernel/initrd-2.6.27.21-0.1-xen --ramdisk
 true
...
Generating manifest /tmp/initrd-2.6.27.21-0.1-xen.manifest.xml

euca-upload-bundle -b example_rd_bucket -m
/tmp/initrd-2.6.27.21-0.1-xen.manifest.xml
...
Uploaded image as
example_rd_bucket/initrd-2.6.27.21-0.1-xen.manifest.xm

euca-register
example_rd_bucket/initrd-2.6.27.21-0.1-xen.manifest.xml
```

```
IMAGE eri-XXXXXXXX
```

Where the returned value `eri-XXXXXXXX` is the unique ID of the registered ramdisk image.

Add a Root Filesystem

When you add a root filesystem to Walrus, you bundle the root filesystem file, upload the file to a bucket in Walrus that you name, and then register the root filesystem with Eucalyptus. The bundle operation can include a registered ramdisk (ERI ID) and a registered kernel (EKI ID). The resulting image will associate the three images.

You can also bundle the root file system independently and associate the ramdisk and kernel with the resulting EMI at run time. For more information, see *Associate a Kernel and Ramdisk*.

To add a root filesystem to Walrus:

Use the following three commands:

```
euca-bundle-image -i <root_filesystem_file>
euca-upload-bundle -b <root_filesystem_file_bucket> -m
/tmp/<root_filesystem_file>.manifest.xml
euca-register <root_filesystem_file_bucket>/<root_filesystem_file>.manifest.xml
```

For example:

```
euca-bundle-image -i euca-fedora-10-x86_64/fedora.10.x86-64.img
--ramdisk eri-722B3CBA --kernel eki-5B3D3859
...
Generating manifest /tmp/fedora.10.x86-64.img.manifest.xml

euca-upload-bundle -b example_rf_bucket -m
/tmp/fedora.10.x86-64.img.manifest.xml
...
Generating manifest /tmp/fedora.10.x86-64.img.manifest.xml

euca-register example_rf_bucket/fedora.10.x86-64.img.manifest.xml
IMAGE  emi-XXXXXXXX
```

Where the returned value `emi-XXXXXXXX` is the unique ID of the registered machine image.

Browse and Install Images from EuStore

Eucalyptus provides a resource - called EuStore - that contains images that you can download and install. This task explains how to browse and install images from EuStore.

To browse and install an image from EuStore:

1. Find an image on EuStore:

```
eustore-describe-images
```

This command returns a list of images available from the EuStore. For example:

```
0400376721 fedora      x86_64  starter        kvm               Fedora 16
x86_64 - SELinux / iptables disabled. Root disk of 4.5G. Root user enabled.
2425352071 fedora      x86_64  starter        kvm               Fedora 17
x86_64 - SELinux / iptables disabled. Root disk of 4.5G. Root user enabled.
```

```
1107385945 centos     x86_64  starter       xen, kvm, vmware  CentOS 5 1.3GB
 root, Hypervisor-Specific Kernels
3868652036 centos      x86_64  starter        kvm               CentOS 6.3
x86_64 - SELinux / iptables disabled. Root disk of 4.5G. Root user enabled.
1347115203 opensuse    x86_64  starter        kvm              OpenSUSE 12.2
 x86_64 - KVM image. SUSE Firewall off. Root disk of 2.5G. Root user enabled.
 Working with kexec kernel and ramdisk. OpenSUSE minimal base package set..
.
.
.
```

For additional information regarding the images on eustore (for example, who is the maintainer of the image), use the -v option:

```
# eustore-describe-images -v
0400376721 fedora      x86_64  starter        kvm               Fedora 16
x86_64 - SELinux / iptables disabled. Root disk of 4.5G. Root user enabled.
     20121107181713      d13e-1e35   fedora-based
olivier.renault@eucalyptus.com
2425352071 fedora      x86_64  starter        kvm               Fedora 17
x86_64 - SELinux / iptables disabled. Root disk of 4.5G. Root user enabled.
     20121107181713      6369-6e28   fedora-based
olivier.renault@eucalyptus.com
1107385945 centos     x86_64  starter       xen, kvm, vmware  CentOS 5 1.3GB
 root, Hypervisor-Specific Kernels
     20120517102326      84ae-59db   centos-based
images@lists.eucalyptus.com
3868652036 centos      x86_64  starter        kvm               CentOS 6.3
x86_64 - SELinux / iptables disabled. Root disk of 4.5G. Root user enabled.
     20121107181713      48df-52d4   centos-based
olivier.renault@eucalyptus.com
1347115203 opensuse    x86_64  starter        kvm              OpenSUSE 12.2
 x86_64 - KVM image. SUSE Firewall off. Root disk of 2.5G. Root user enabled.
 Working with kexec kernel and ramdisk. OpenSUSE minimal base package set..
     20121120130646      a981-db13   opensuse-based
lester.wade@eucalyptus.com
```

2. Pick an available image from the returned list and note the image ID. For this example, we will choose:

```
3868652036 centos      x86_64  starter        kvm               CentOS 6.3
x86_64 - SELinux / iptables disabled. Root disk of 4.5G. Root user enabled.
```

3. Install the image from EuStore using the eustore-install-image command. For this example, we only need to specify the image ID and the name of a bucket (the bucket will be created if it doesn't already exist):

   Note: Some images may require additional parameters (for example, that you specify a kernel type with the -k option). Please see the eustore-install-image topic in the Eucalyptus Command Line Reference for more information.

```
eustore-install-image -b centos-testbucket -i 3868652036 -k kvm
```

This command performs a number of tasks for you, including downloading the image from the central Eucalyptus image store and installing the image on your own Eucalyptus private cloud. The output from this command will look similar to the following example:

```
Downloading Image :  CentOS 6.3 x86_64 - SELinux / iptables disabled. Root
disk of 4.5G. Root user enabled.
0-----1-----2-----3-----4-----5-----6-----7-----8-----9-----10
##############################################################

Checking image bundle
Unbundling image
going to look for kernel dir : kvm-kernel
Bundling/uploading ramdisk
Checking image
Compressing image
Encrypting image
Splitting image...
Part: initrd-2.6.32-279.14.1.el6.x86_64.img.part.00
Generating manifest
/tmp/olEuG_/initrd-2.6.32-279.14.1.el6.x86_64.img.manifest.xml
Checking bucket: centos-testbucket
Creating bucket: centos-testbucket
Uploading manifest file
Uploading part: initrd-2.6.32-279.14.1.el6.x86_64.img.part.00
Uploaded image as
centos-testbucket/initrd-2.6.32-279.14.1.el6.x86_64.img.manifest.xml
centos-testbucket/initrd-2.6.32-279.14.1.el6.x86_64.img.manifest.xml
eri-064B387A
Bundling/uploading kernel
Checking image
Compressing image
Encrypting image
Splitting image...
Part: vmlinuz-2.6.32-279.14.1.el6.x86_64.part.00
Generating manifest /tmp/olEuG_/vmlinuz-2.6.32-279.14.1.el6.x86_64.manifest.xml
Checking bucket: centos-testbucket
Uploading manifest file
Uploading part: vmlinuz-2.6.32-279.14.1.el6.x86_64.part.00
Uploaded image as
centos-testbucket/vmlinuz-2.6.32-279.14.1.el6.x86_64.manifest.xml
centos-testbucket/vmlinuz-2.6.32-279.14.1.el6.x86_64.manifest.xml
eki-A4D6398A
Bundling/uploading image
Checking image
Compressing image
Encrypting image
Splitting image...
Part: centos-6.3-x86_64.part.00
Part: centos-6.3-x86_64.part.01
Part: centos-6.3-x86_64.part.02
Part: centos-6.3-x86_64.part.03
Part: centos-6.3-x86_64.part.04
Part: centos-6.3-x86_64.part.05
Part: centos-6.3-x86_64.part.06
Part: centos-6.3-x86_64.part.07
Part: centos-6.3-x86_64.part.08
Part: centos-6.3-x86_64.part.09
Part: centos-6.3-x86_64.part.10
Part: centos-6.3-x86_64.part.11
Part: centos-6.3-x86_64.part.12
Part: centos-6.3-x86_64.part.13
Part: centos-6.3-x86_64.part.14
```

```
Part: centos-6.3-x86_64.part.15
Part: centos-6.3-x86_64.part.16
Part: centos-6.3-x86_64.part.17
Part: centos-6.3-x86_64.part.18
Part: centos-6.3-x86_64.part.19
Generating manifest /tmp/olEuG_/centos-6.3-x86_64.manifest.xml
Checking bucket: centos-testbucket
Uploading manifest file
Uploading part: centos-6.3-x86_64.part.00
Uploading part: centos-6.3-x86_64.part.01
Uploading part: centos-6.3-x86_64.part.02
Uploading part: centos-6.3-x86_64.part.03
Uploading part: centos-6.3-x86_64.part.04
Uploading part: centos-6.3-x86_64.part.05
Uploading part: centos-6.3-x86_64.part.06
Uploading part: centos-6.3-x86_64.part.07
Uploading part: centos-6.3-x86_64.part.08
Uploading part: centos-6.3-x86_64.part.09
Uploading part: centos-6.3-x86_64.part.10
Uploading part: centos-6.3-x86_64.part.11
Uploading part: centos-6.3-x86_64.part.12
Uploading part: centos-6.3-x86_64.part.13
Uploading part: centos-6.3-x86_64.part.14
Uploading part: centos-6.3-x86_64.part.15
Uploading part: centos-6.3-x86_64.part.16
Uploading part: centos-6.3-x86_64.part.17
Uploading part: centos-6.3-x86_64.part.18
Uploading part: centos-6.3-x86_64.part.19
Uploaded image as centos-testbucket/centos-6.3-x86_64.manifest.xml
centos-testbucket/centos-6.3-x86_64.manifest.xml
Installed image: emi-233637E1
```

Note the last line in the output, which provides the image ID for the image you just installed from the euca store. In this example, the image ID is emi-233637E1.

4. Verify the image was installed on your Eucalyptus cloud. To do this, use the euca-describe-images command, which returns a list of the available images on your Eucalyptus cloud:

```
euca-describe-images | grep centos-testbucket
```

This command will return output similar to the following example:

```
IMAGE   eki-A4D6398A
centos-testbucket/vmlinuz-2.6.32-279.14.1.el6.x86_64.manifest.xml
345590850920    available   public      x86_64  kernel
instance-store
IMAGE   eri-064B387A
centos-testbucket/initrd-2.6.32-279.14.1.el6.x86_64.img.manifest.xml
345590850920    available   public      x86_64  ramdisk
instance-store
IMAGE   emi-233637E1    centos-testbucket/centos-6.3-x86_64.manifest.xml
345590850920    available   public      x86_64  machine eki-A4D6398A
eri-064B387A        instance-store
```

Note the ID of the last image in the output -

```
emi-233637E1
```

- matches that of the image we installed from EuStore.

The image has been successfully downloaded from EuStore and installed on your Eucalyptus cloud.

You can now run an instance from this image and connect to it using SSH.

Associate a Kernel and Ramdisk

There are three ways that you can associate a kernel and ramdisk with an image.

Tip: This only applies to Linux images. Windows images do not use kernels or ramdisks.

- You can associate a specific kernel and ramdisk identifier with an image at the `euca-bundle-image` step.

```
euca-bundle-image -i <vm_image_file> --kernel <eki-XXXXXXXX> --ramdisk
<eri-XXXXXXXX>
```

- You can specific kernel and ramdisk at instance run time as an option to `euca-run-instances` command.

```
euca-run-instances --kernel <eki-XXXXXXXX> --ramdisk <eri-XXXXXXXX>
<emi-XXXXXXXX>
```

- An administrator can set default registered kernel and ramdisk identifiers that will be used if a kernel and ramdisk are unspecified by either of the other options.

Modify an Image

You might find that existing images in your cloud don't meet your needs.

To modify an existing image:

1. Create a mount point for your image.

```
mkdir temp-mnt
```

2. Mount the image.

```
mount -o loop <image_name> temp-mnt
```

3. Make procfs, dev and sysfs available in your chroot environment.

```
mkdir -p temp-mnt/proc
mkdir -p temp-mnt/sys
mkdir -p temp-mnt/dev
mount -o bind /proc temp-mnt/proc
mount -o bind /sys temp-mnt/sys
mount -o bind /dev temp-mnt/dev
```

4. You now have the image under `temp-mnt` and you can copy over what you want into it. If you want to install packages into it, you have few options:

- `chroot temp-mnt` and use `apt-get`, `yum`, or `zypper` to install what you want.
- Instruct the package manager program to use a different root (for example both dpkg and rpm uses the --root option)

5. Unmount the drive.

```
umount temp-mnt
```

You now have an image with your modifications. You are ready to *add the image* to Eucalyptus.

Remove an Image

When a new image is uploaded to Eucalyptus, Eucalyptus saves the bundle and the image manifest to a bucket in Walrus. This bucket is set in the Eucalyptus property `walrus.storagedir`. The default value for this property is `/var/lib/eucalyptus/bukkits`.

When you register an image, Eucalyptus creates the actual image file. Both the image manifest and the bundle must remain intact to run as an instance.

Eucalyptus stores images in the path set

To delete an image fully, you must deregister the image and delete the bundle. To successfully remove an image and associated bundle files:

1. Find the image you want to remove.

```
euca-describe-images
IMAGE   emi-E533392E    alpha/centos.5-3.x86-64.img.manifest.xml
965590394582
available   public   i386    machine eki-345135C9    eri-C4F135BC
instance-store
IMAGE   emi-623C38B0  alpha/ubuntu.9-04.x86-64.img.manifest.xml   965590394582

available   public   i386    machine eki-E6B13926    eri-94DB3AB9
instance-store
```

2. Note the image file name (for example, emi-623C38B0).
3. Deregister the image.

```
euca-deregister emi-623C38B0
IMAGE   emi-623C38B0
```

4. Deregister the image again to delete the image and remove it from the available images list.

```
euca-deregister emi-623C38B0
IMAGE   emi-623C38B0
```

5. Delete the bundle.

```
euca-delete-bundle -b alpha -p ubuntu.9-04.x86-64.img
```

Tip: If you accidentally try to delete a bundle for a second time, you might see an error message: `problem parsing: /tmp/centos.5-3.x86-64.img.manifest.xml`. This error only displays if you try to delete a bundle that no longer exists.

When you have finished these steps, display all images to confirm that the image was removed.

```
euca-describe-images
IMAGE   emi-E533392E    alpha/centos.5-3.x86-64.img.manifest.xml    965590394582
available   public   i386    machine eki-345135C9    eri-C4F135BC
instance-store
```

Creating an Image

You can create your own image if you to deploy an instance that is customized for your specific use. For example, you can create an EMI to deploy a LAMP server, MySQL database server, or a Postgres database server. All you need is some familiarity with the creation process.

About Linux Images

A Linux image is a root partition of a Linux installation. We use the convention followed by EC2 images:

- The first partition is the root partition (where the EMI is attached)
- The second is the ephemeral partition (for additional storage)
- The third is the swap partition

Any image you create must conform to these conditions (for example, `/etc/fstab` must follow this convention). Typically, the disk provided by Eucalyptus (as well as EC2) is the first SCSI disk (sda). However, you can customize it to be, for example, the first IDE disk (hda) or the first virtual disk on a paravirtual machine (xvda).

About Windows Images

Eucalyptus supports a licensed version of Windows images. Before you bundle a Windows image, you must have a valid version of Windows OS installed on your hypervisor and the Eucalyptus Windows Integration Service installed in the created Windows VM. The following sections detail how to perform these tasks.

Eucalyptus is compatible with images created from licensed versions of Windows Server 2003 R2, Windows Server 2008, Windows Server 2008 R2, and Windows 7. Windows OS is sensitive to physical and virtual hardware changes made after installation. We recommend that you create any Windows image on the specific hypervisor (Xen, KVM, or VMware) where you plan to run instances of that Windows image. Eucalyptus does not support running a Windows image across different hypervisors.

Before you begin, you will need the following room for a blank disk file:

- 8GB minimum, for Windows Server 2003 R2 (32-bit and 64-bit)
- 15GB minimum for Windows Server 2008 (32-bit and 64-bit), Windows Server 2008 R2 (64-bit), and Windows 7 (32-bit and 64-bit)

Tip: Eucalyptus implements ephemeral disks by which a running Windows instances are allocated an additional disk space based on the instance type. For example, if you assign an instance type 15 GB of disk and the registered Windows EMI is 8 GB, 7 GB of disk spaces are ephemeral disks accessible under `D:\`. If your Windows application uses scratch space most of time, we recommend keeping the Windows EMI small because it takes longer to launch bigger EMIs.

The windows image is a root file system that has no kernel and ramdisk associated with it. After creating your own Windows images, you can bundle, upload, and register it with Eucalyptus.

Hypervisor and OS Information

The following sections detail how to create an image based on the hypervisor and image OS you wish to use. To create an image using Windows, see *Create Windows Image (KVM)*. To create an image using Linux with either KVM or Xen, see *Create Linux Image*.

Create Windows Image (KVM)

We recommend that you perform this task on a node controller (NC), or a host running the same Linux distributions and hypervisors as your NCs. If you are creating the Windows image on a machine currently running as a NC, terminate all running instances and stop the NC. To stop the NC, enter:

```
service eucalyptus-nc stop
```

A template file that closely matches those that Eucalyptus generates at VM instantiation time is located at `/usr/share/eucalyptus/doc/libvirt-kvm-windows-example.xml`. We recommend that you review the file to acquaint yourself with its contents, noting required files, bridges, and resources. For more information about configuring the libvirt.xml file, go to the *Domain XML format* page in the libvirt documentation.

To create an image from a Windows OS in VMware you will need one network interface and one disk.

Note: If you are using VMware, make sure that the Windows VM uses the LSI Logic Parallel driver as the SCSI controller. For some Windows versions, this is not the default SCSI controller in the VM setting.

Install Base Windows OS

The first task for creating a Windows image is installing a base Windows operating system (OS). To install a base Windows OS using KVM:

1. Log in to the stopped NC server or a host that runs the same hypervisor as the NCs.
2. Create a blank disk file. Specify your Windows VM image name using the parameter `of`.

```
dd if=/dev/zero of=windows.<image_name>.img bs=1M count=1 seek=16999
```

   Important: Your image name must start with the word, `windows` (all lower-case).

3. Create a floppy and secondary blank disk to be attached to the image later, in order to test paravirtualization drivers

```
dd if=/dev/zero of=floppy.img \
 bs=1k count=1474
 dd if=/dev/zero of=secondary.img \
 bs=1M count=1 seek=1000
```

4. Copy all of the .img and .iso files to the `/var/lib/libvirt/images/` directory.
5. Copy the `libvirt-kvm-windows-example.xml` file to your working directory and rename it to `libvirt-kvm-windows.xml`.

```
 cp /usr/share/eucalyptus/doc/libvirt-kvm-windows-example.xml
/var/lib/libvirt/images/libvirt-kvm-windows.xml
```

   or

```
 cp /usr/share/eucalyptus/doc/libvirt-xen-windows-example.xml
/var/lib/libvirt/images/libvirt-xen-windows.xml
```

6. Open the new `libvirt-kvm-windows.xml` file and provide fully qualified paths to the VM image file and iso. Make sure that the name of the bridge is the same as the one used by the hypervisor on which you are creating the Windows image.

   Your file should look similar to the following example:

```
<domain type='kvm'>
   <name>your-domain-name-here</name>
    <os>
    <type>hvm</type>
    <boot dev='cdrom'/>
    </os>
    <features>
        <acpi/>
    </features>
    <memory>524288</memory>
    <vcpu>1</vcpu>
    <devices>
        <emulator>/usr/libexec/qemu-kvm</emulator>
        <disk type='file'>
            <source file='/var/lib/libvirt/images/windows_2003.img'/>
            <target dev='hda'/>
        </disk>
        <!-- <disk type='file' device='disk'>
             <source file='fully_qualified_path_to_secondary_disk'/>
             <target dev='vda' bus='virtio'/>
        </disk>
        <disk type='file' device='floppy'>
             <source file='fully_qualified_path_to_floppy_disk'/>
             <target dev='fda'/>
        </disk> -->
        <disk type='file' device='cdrom'>
            <source
file='/var/lib/libvirt/images/en_win_srv_2003_r2_enterprise_with_sp2_cd1_x13-05460.iso'/>

            <target dev='hdc'/>
            <readonly/>
        </disk>
        <interface type='bridge'>
            <source bridge='br0'/>
            <model type='rtl8139'/>
        </interface>
        <!--<interface type='bridge'>
            <source bridge='br0'/>
            <model type='virtio'/>
        </interface> -->
        <graphics type='vnc' port='-1' autoport='yes' listen='0.0.0.0'/>
    </devices>
</domain>
```

7. Start the VM.

```
cd /var/lib/libvirt/images
virsh create libvirt-kvm-windows.xml
```

8. Connect to the virtual console using the VNC client of your choice. On the NC, check the display number that has been allocated by looking at the process table (`ps axw | grep vnc`). For example, if the display number is 0, then connect to the NC using the VNC client:

```
vinagre <machine-hosting-vm>:0
```

9. Follow the standard Windows installation procedure until the VM has completed installing Windows.

   Tip: On some hosts, the VNC's display number will change when an image restarts. Use `ps` to find the current number.

10. Run `virsh list` to display the domain name.
11. Shut down the Windows VM you have just created. The easiest way to shutdown your VM is to use the `virsh destroy` command, as shown:

```
virsh destroy <domain_name>
```

To install the base Windows operating system using VMware, create a new VM using the VMware vSphere Client. Install Windows on the VM following standard VMware procedures, and install VMware Tools.

Install Eucalyptus Windows Integration

To install the Eucalyptus Windows Integration Service:

1. Download the most recent version of the Windows Image Preparation Tool from *http://downloads.eucalyptus.com/software/tools/windows-prep/* to `/var/lib/libvirt/images` on your NC or on the host running the vSphere client.
2. Open the libvirt-kvm-windows-example.xml file you used in the previous section and make the following edits:
   - Comment out the lines of XML code directing the hypervisor to boot the Windows image from the CDROM.
   - Change the text so that `windows-prep-tools-latest.iso` replaces the Windows .iso image and is mounted as `cdrom`.
   - Enter the path to the secondary disk file you created in the previous task.
   - Uncomment the lines that direct attachment of a floppy disk, secondary disk, and secondary network interface.

   Tip: If you plan on using virtio networking for instances (via USE_VIRTIO_NET option on node controllers), uncommenting the virtio interface in the xml is mandatory

   Your finished file should look similar to the following example:

```
<domain type="kvm">
   <name>eucalyptus-windows</name>
    <os>
    <type>hvm</type>
    <!-- <boot dev='cdrom'/> -->
    </os>
    <features>
        <acpi/>
    </features>
    <memory>524288</memory>
    <vcpu>1</vcpu>
    <devices>
        <emulator>/usr/libexec/qemu-kvm</emulator>
        <disk type='file'>
            <source file='/var/lib/libvirt/images/windows_2003.img'/>
            <target dev='hda'/>
        </disk>
        <disk type='file' device='disk'>
             <source file='/var/lib/libvirt/images/secondary.img'/>
             <target dev='vda' bus='virtio'/>
        </disk>
        <disk type='file' device='floppy'>
             <source file='/var/lib/libvirt/images/floppy.img'/>
             <target dev='fda'/>
        </disk>
        <disk type='file' device='cdrom'>
            <source
file='/var/lib/libvirt/images/windows-preps-tools-latest.iso'/>
            <target dev='hdc'/>
            <readonly/>
        </disk>
        <interface type='bridge'>
            <source bridge='br0'/>
            <model type='rtl8139'/>
        </interface>
        <interface type='bridge'>
            <source bridge='br0'/>
            <model type='virtio'/>
        </interface>
        <graphics type='vnc' port='-1' autoport='yes' listen='0.0.0.0'/>
    </devices>
</domain>
```

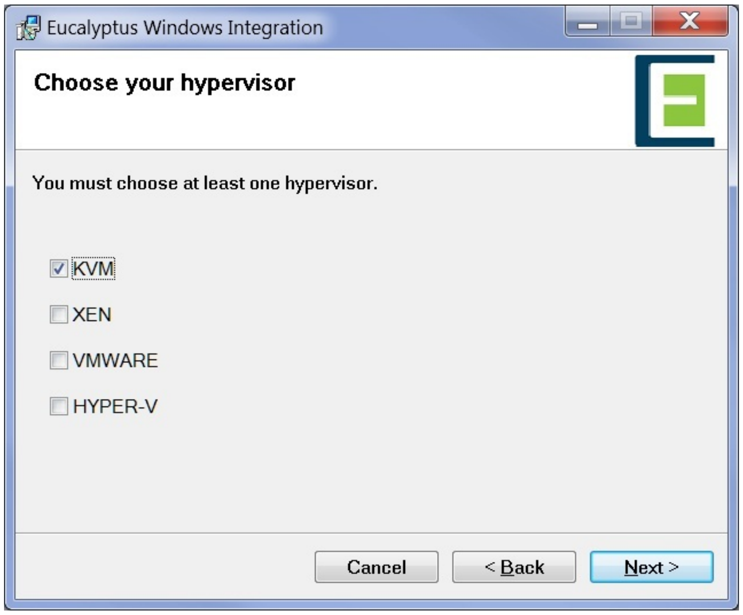
3. Start the VM.

```
cd /var/lib/libvirt/images/
virsh create libvirt-hypervisor-windows.xml
```

4. Start the VM with the newly modified libvirt xml file:

```
virsh create libvirt-kvm-windows.xml
```

5. For VMware, use the VMware vSphere client to upload the ISO file to the VSphere datastore. Attach the `windows-prep-tools-latest.iso` to the Windows VM.
6. Log in to Windows and find the Eucalyptus installation files in the CDROM drive.
   - For Windows Server 2003 R2, run `setup.exe`. This automatically installs the .NET framework 2.0, which is not bundled in Server 2003 R2.
   - For all other versions, run `EucalyptusWindowsIntegration.msi`. (setup.exe will automatically install .NET framework 2.0, which is not bundled in Server 2003 R2).
7. In the **Choose your hypervisor** step, select **KVM** and then click **Next**.



You must choose at least one hypervisor.

[✓] KVM
[ ] XEN
[ ] VMWARE
[ ] HYPER-V

   Click **Next** and continue until the end of installation.

8. Reboot the Windows VM
9. Open the Windows device manager and check that the following drivers are found for each device.
   - Floppy disk drive
   - Disk drivers: Red Hat VirtIO SCSI Disk Device
   - SCSI and RAID controllers: Red Hat VirtIO SCSI controller
   - Network adapters: Red Hat VirtIO Ethernet Adapter

   If the correct drivers are not found, question marks display on the devices. To install the devices, do the following:

   - Right-click on the devices in question and select **Update Drivers** to open the New Hardware Wizard.
   - When the new hardware wizard asks if Windows update is to be connected, click **No, not this time**.
   - Choose **Install software automatically (recommended)**.
   - If a confirmation popup message displays, click **Continue**.

You are now ready to *Configure Active Directory*.

Configure Active Directory

The Eucalyptus Integration service lets an enterprise with existing Active Directory domains attach Windows instances to the domains and control access to these instances using the existing AD user database. Users can log into the instance either using their domain credentials or the Administrator's password generated with the `euca-get-password` command.

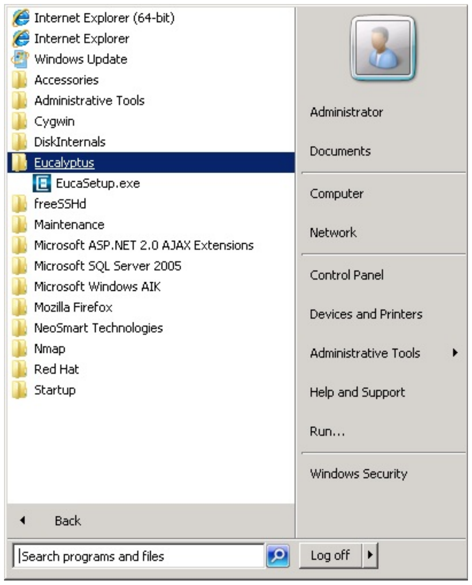
Because AD technology is tightly integrated with domain name service (DNS), the default name server contacted by the instance must be able to resolve the AD address as a proper domain controller. You can do this for all networking modes except System, by configuring the following line the CC's eucalyptus.conf file:

```
VNET_DNS=<domain_controller_IP_address>
```

If there is no such pre-existing DNS set-up or your networking mode is System, you might need to change the VM's network interface so that the preferred DNS server points to the domain controller.

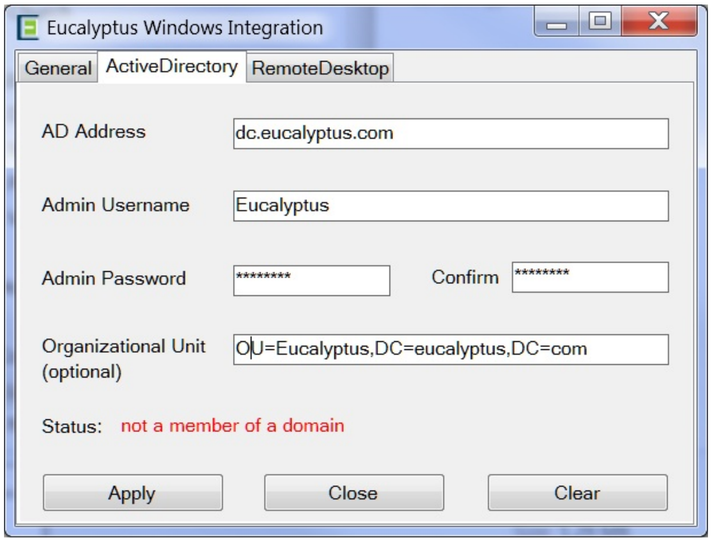
To set up Active Directory:

1. Click **Windows Programs** > **Eucalyptus** > **Eucalyptus Setup**.



The **Eucalyptus Windows Integration** popup displays.

2. Click the **Active Directory** tab in the Eucalyptus Windows Integration window and enter the following information:



AD Address: dc.eucalyptus.com

Admin Username: Eucalyptus

Admin Password: ******** Confirm: ********

Organizational Unit (optional): OU=Eucalyptus,DC=eucalyptus,DC=com

Status: not a member of a domain

- Enter the domain name of the existing Active Directory domain controller in the **AD Address** field.
- Enter the administrator username in the **Admin Username** field. We recommend using a generic user account that has permission to join a computer to a domain or a specific organizational unit.
- Enter and confirm the password in the **Admin Password** field. Note that the Admin username and password are required to join an instance to an Active Directory. When launched in Eucalyptus, these properties will be deleted as soon as the instance joins (or fails to join) the domain.
- Optionally, enter an organizational unit in the **Organizational Unit** field. This specifies a container that the instances launched from this image will be attach to.

Tip: If the values entered in this section are incorrect, the launched instances will fail to join the domain. We recommend that you verify the information by manually joining a computer to a domain using the same information that you entered in this step. You may first log in the launched instance using the administrator password (`euca-get-passcword`) and manually join the domain for verification.

3. Click **Apply**.

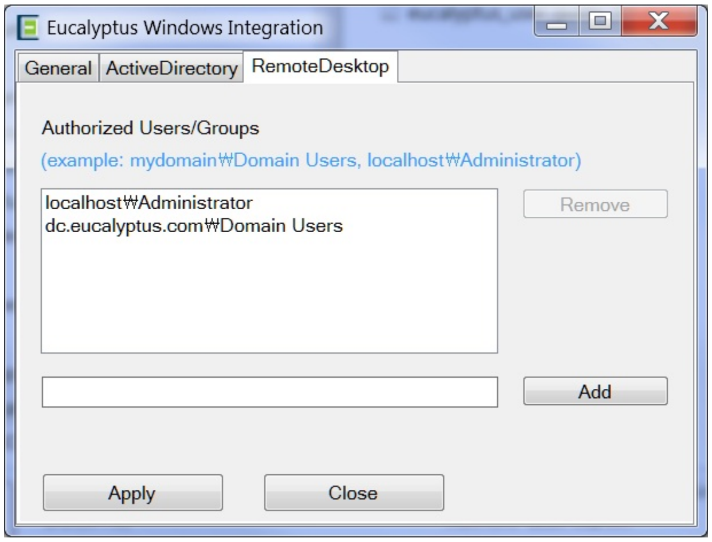
You are now ready to *Configure Remote Desktop*.

Configure Remote Desktop

Domain users or groups require remote desktop permission to log into an instance. By default, only the local administrator has the remote desktop permission. The Eucalyptus Integration Service provides a way to grant remote desktop permission to additional domain users or groups.

To configure remote desktop permission:

1. Open the **Eucalyptus Windows Integration** popup (**Windows Programs** > **Eucalyptus** > **Eucalyptus Setup**).
2. Click the **RemoteDesktop** tab

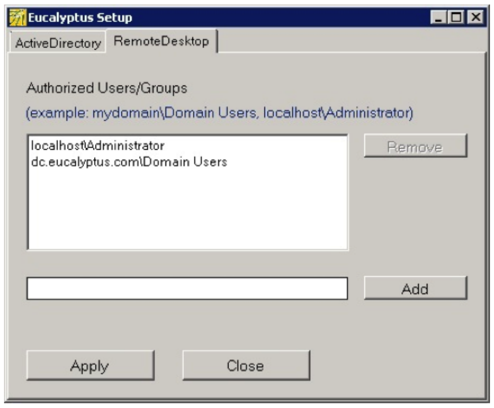


Authorized Users/Groups
(example: mydomain₩Domain Users, localhost₩Administrator)

localhost₩Administrator
dc.eucalyptus.com₩Domain Users

The names of authorized domain users and groups display in the **Authorized User/Groups** field. By default, only the local administrator is listed as authorized.

3. In the text field below the list, enter a user and group account name in the format `[DOMAIN]\[USER or GROUP]`. If you add a new local user or local group, prepend the account name `localhost\` instead of the domain name.



Authorized Users/Groups
(example: mydomain\Domain Users, localhost\Administrator)

localhost\Administrator
dc.eucalyptus.com\Domain Users

4. Click **Add**.
5. Repeat for all user/groups that you want to add.
6. Click **Apply**.
   When the instance launches, the members of the groups you added can log in to the instance through remote desktop.

You are now ready to *Run Sysprep*.

Run Sysprep

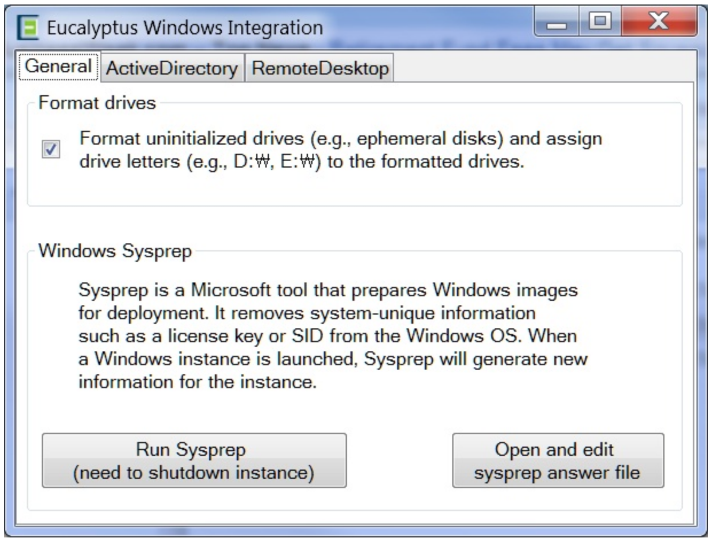
Sysprep is a Microsoft tool for deploying multiple Windows operating systems in an enterprise. Running Sysprep removes system-specific information such as security ID (SID) from the Windows OS before you clone an image. Sysprep then re-initializes the OS after the image is cloned and started on multiple computers. Use Sysprep to prepare images when you use Microsoft Key Management Service to activate license keys. Also, use Sysprep when your Windows systems are attached to Active Directory.

In Eucalyptus, you can run Sysprep before you bundle images with the `euca-bundle-image` or `euca-bundle-instance` commands.

Note: The Eucalyptus Integration Service supports Sysprep for Windows Server 2008, Windows Server 2008 R2, and Windows 7.

To configure and run Sysprep:

1. Open the **Eucalyptus Windows Integration** popup (**Windows Programs** > **Eucalyptus** > **Eucalyptus Setup**).



Format drives

☑ Format uninitialized drives (e.g., ephemeral disks) and assign drive letters (e.g., D:\, E:\) to the formatted drives.

Windows Sysprep

Sysprep is a Microsoft tool that prepares Windows images for deployment. It removes system-unique information such as a license key or SID from the Windows OS. When a Windows instance is launched, Sysprep will generate new information for the instance.

2. Click the **General** tab.
3. Ensure that **Format uninitialized drives** is checked

4. If you want to edit the Sysprep answer file, click the **Open and change** button in the General tab. Otherwise skip this step.
5. Click the **Run Sysprep** button.
   Sysprep starts.
6. After Sysprep is complete, close the application and shutdown the Windows VM using the Windows Programs menu.

You are now ready to *Convert the VMDK to an Image*.

Convert the VMDK to an Image

After your Windows VM image is ready on ESX/ESX-i/VCenter, use the `euca_imager` command to download the image and convert it to the format Eucalyptus uses. The `euca_imager` command converts a remote VMDK located on a datastore into a disk image on the local disk on the CC/VMware Broker machine.

1. Log into the CC/VMware Broker machine, set the required environment variables, and source your `eucarc` file.

```
export VDDK_HOME=/opt/packages/vddk/
export EUCALYPTUS=/
export LD_LIBRARY_PATH=/opt/packages/vddk/lib/vmware-vix-disklib/lib64/
source ~/.euca/eucarc
```

2. Enter the euca_imager command with appropriate parameters to extract and convert the VMDK to a raw disk image. Parameters are described in the following table:

Parameter	Description
`debug`	Enables `euca_imager` debugging output
`in`	https URL for the vCenter Server or ESX/ESXi machine
`out`	User-defined name of disk
`login`	Name used to access the VMware machine
`password`	Password used to access the VMware machine
`vsphere-vmdk`	Go to vSphere client, find the virtual machine that you have installed (make sure it is in `powered off` state), and select the **Hard disk 1** resource. There will be a **Disk File** field for the VMDK, and a similar field on the **Option** tab for the VMX file.
`vsphere-vmx`	Same description as vsphere-vmdk parameter. This is optional if you're contacting an ESX node directly, but may be required when contacting vCenter.
`vsphere-datacenter`	Specified name of the datacenter. This is optional if you're contacting an ESX node directly, but may be required when contacting vCenter.

The following example shows euca_imager with the appropriate parameters as described above converting a VMDK to a raw disk image. Note that the prefix of the `out` parameter must begin with the word `windows`.

```
/usr/lib/eucalyptus/euca_imager debug=yes convert in=
https://192.168.7.198 out=windows.disk login=Administrator
password=password
vsphere-vmdk="[datastore1]WindowsVM/WindowsVM.vmdk"
```

```
vsphere-datacenter=DataCenter1
vsphere-vmx="[datastore1]WindowsVM/WindowsVM.vmx"
```

You are now ready to *Add Image to Eucalyptus*.

Add Image to Eucalyptus

To enable an image as an executable entity, you must bundle and upload the Windows disk image to Walrus, and then register the uploaded image with Eucalyptus.

Run the following command to bundle, upload, and register your Windows disk image:

```
euca-bundle-image -i <vm_image_file>
euca-upload-bundle -b <image_bucket> -m /tmp/<vm_image_file>.manifest.xml
euca-register <image_bucket>/<vm_image_file>.manifest.xml
```

Your Windows image is now ready to run as an instance.

After you register the image, Walrus decrypts the image bundle. This process might take a few minutes for a large Windows image to be decrypted. For example, a 10G image requires that you wait about 10 minutes before you launch the instance. You can use the *`euca-describe-bundle-tasks`* command line utility to check the status of active bundle tasks.

Create Linux Image

Important: Before you begin, make sure that your hypervisor is installed and properly functioning for the account you are going to use.

To create a root filesystem to be used with Eucalyptus using Xen or KVM and an ISO image of the guest OS you want to install:

1. Download the ISO image of the distro (guest OS) you want to install.
2. Create a virtual disk with the desired size. For example, to create a 4 GiB disk:

```
dd if=/dev/zero of=<image_name> bs=1M count=4096
```

3. If you are using Xen, complete the following steps:

   a) Create a configuration file (for example, `xen.cfg`) to boot off the ISO image. The file should look like the following:

```
name = "bootFromISO"
kernel = "/usr/lib/xen/boot/hvmloader"
memory = 1024
builder = "hvm"
device_model = "/usr/lib64/xen/bin/qemu-dm"
boot = "d"
disk = ['file:PATH_TO_ISO,hdc:cdrom,r',
'file:PATH_TO/<image_name>,sda,w']
vif = ['mac=00:01:01:00:00:03, bridge=xenbr0']
dhcp="on"
vnc = 1
vncdisplay = 7
pae = 1
```

   Important: You must modify `device_model` and `kernel` since they depend on where the distribution puts such files. Look into the Xen package file list. Also be sure that all the options are correct for your needs.

b) Start the domU.

```
xen create xen.cfg
```

c) Connect with a VNC viewer. For example:

```
xvncviewer localhost:7
```

4. If you are using KVM, start KVM using the CDROM as the boot device and new image as the disk.

```
/usr/libexec/qemu-kvm -cdrom iso_image -drive
if=scsi,file=<image_name>,boot=off
```

Note: You need to have a SCSI as a bus because Eucalyptus expects the disk to be in `/dev/sda`.

5. Install the distro as you would normally would. Do not use `lvm` or `md`. Install everything on one partition. Remember the partition on which the distro is installed.

   Important: Most distributions use the MAC address to associate each network interface with an interface name. To ensure that each instance started from the image sees the appropriate network interfaces as `eth0` etc, you must remove this association. Refer to your distribution's documentation for information on how to do this. On Debian and Ubuntu, `/etc/udev/rules.d/*net*` may be relevant. On CentOS and RHEL, `/etc/network-scripts/ifcfg-eth0` may be relevant. Failure to remove this association may leave your instance unable to connect to the network.

6. If you are using Xen, stop the domU.
7. Run `parted` to find out the starting block and the block size of the root file system, in bytes.

```
parted <image_name>
```

8. Change the units shown by `parted`to blocks.

```
(parted) u
Unit? [compact]? b
```

9. Print the current partition table using `p`.
   Note the start block and block size for the partition to which you will extract the file system. This example uses a start block of `32256` and a block size of `1024000`.
10. Extract the file system.

```
dd if=<image_name> of=rootfs.img bs=1 skip=32256 count=1024000
```

Tip: If this process is running slowly, you can increase the block size. Set `bs` to a power of two, and divide `skip` and `count` by the same number. For example, using a block size of 512, the above becomes `dd if=<image_name> of=rootfs.img bs=512 skip=63 count=2000`.

You now have a root filesystem. Make sure that it is compatible with the kernel/initrd you are using in your cloud environment. In particular you may want to be sure you have the modules of the kernel you are going to use. Then you are ready to *add the image* to Eucalyptus.

Create Linux Image (VMware)

To create a root filesystem to be used with Eucalyptus using VMware and an ISO image of the guest OS you want to install:

1. Run an instance of the image.
2. Attach a volume to the instance.

```
euca-attach-volume -i <instance_id> -d <local_device_name> <volume_id>
```

3. Make a file system on the volume and mount it to `/mnt`.

```
mkfs -t ext3 /dev/vdb
mount /dev/vdb /mnt
```

4. SCP your Eucalyptus credential zip file to the instance.

```
scp -i yourkey.private euca2-xxxxxxx-x509.zip root@xxx.xxx.xxx.xxx:/mnt
```

5. SSH into the instance.

```
ssh -i yourkey.private root@xxx.xxx.xxx.xxx
```

6. Install the tools you need.

```
yum install rsync unzip
```

7. Change to the `/mnt` device.

```
cd /mnt
```

8. Unzip your Eucalyptus credentials zip to `/mnt/euca`.

```
unzip euca2-admin-x509.zip -d euca/
```

9. Install any other software you want on the instance so it will be part of your new image.
10. Remove the udev rules.

```
rm /etc/udev/rules.d/70*
```

You now have a root filesystem. Make sure that it is compatible with the kernel/initrd you are using in your cloud environment. In particular you may want to be sure you have the modules of the kernel you are going to use. Then you are ready to *add the image* to Eucalyptus.

Create an EBS-Backed Image

An EBS-backed image (sometimes referred to as a "bfEBS" image) is an image with a root device that is an EBS volume created from an EBS snapshot. An EBS-backed image has a number of advantages, including:

- Faster boot time
- Larger volume size limits
- Changes to the data on the image persist after instance termination

Create an Image File

You can create an EBS-backed EMI from an existing .img file or create your own .img file. One way to create your own .img file is to use virt-install as described below.

Note: If you already have .img file, skip to the *Register an EBS-Backed EMI* section.

Note: Use `virt-install` on a system with the same operating system version and hypervisor as your Node Controller. If you use an image created by virt-install under a different distribution or hypervisor combination, it is likely that it will not install the correct drivers into the ramdisk and the image will not boot on your Node Controller.

To create an EBS-backed image file:

1. On the designated host, use the QEMU disk utility to create a disk image. For example:

   `qemu-img create -f raw bfebs.img 2G`
2. Use `parted` to set the disk label:.

   ```
   parted bfebs.img mklabel msdos
   ```
3. Use virt-install to start a new virtual machine installation using the disk image you just created:

   ```
   virt-install --name rhel6 --ram 1024 --os-type linux --os-variant rhel6 \
   -c /tmp/media.iso --disk \
   path=/tmp/bfebs.img,device=disk,bus=virtio --vnc
   ```
4. Once you have completed the installation, start the virtual machine using virt-manager or other libvirt tool of your choice.
5. Configure the virtual machine by connecting to it and making the following changes:
   a) Comment out the `HWADDR` entry from the `/etc/sysconfig/network-scripts/ifcfg-eth0` file. For example:

   ```
   DEVICE="eth0"
   BOOTPROTO="dhcp"
   #HWADDR="B8:AC:6F:83:1C:45"
   IPV6INIT="yes"
   MTU="1500"
   NM_CONTROLLED="yes"
   ONBOOT="yes"
   TYPE="Ethernet"
   UUID="499c07cc-4a53-408c-87d2-ce0db991648e"
   PERSISTENT_DHCLIENT=1
   ```

   b) Add the following option to the end of `/boot/grub/menu.lst` to get a serial console:

   `console=ttyS0`

   c) Remove the `quiet` option from the kernel parameters and grub menu splash image in the `/boot/grub/menu.lst` file.

   d) Add the following line to the `/etc/sysconfig/network` file to disable the zeroconf route, which can interfere with access to the metadata service:

   `NOZEROCONF=yes`

   e) Edit `/etc/udev/rules.d/70-persistent-net.rules` and remove the entry for the existing NIC.

   f) Copy the Eucalyptus `rc.local` file from https://github.com/eucalyptus/Eucalyptus-Scripts/blob/master/rc.local.

Register an EBS-Backed EMI

Creating an EBS-backed EMI will require initial assistance from a helper instance. The helper instance can be either an instance store or EBS-backed instance and can be deleted when finished. It only exists to help create the initial volume that will be the source of the snapshot behind the EBS-backed EMI.

1. Create and launch a help instance.
2. Create a volume big enough to hold the bootable .img file.

   `euca-create-volume -z <cluster_name> -s <size_in_GB>`

3. **3.** Attach the volume to the helper instance.

   `euca-attach-volume <volume-id> -i <instance-id> -d <device>`
4. Log in to the instance and copy the bootable image to the attached volume by performing one of the following steps:
   - If the bootable image is saved in an http or ftp repository, use `curl` or `wget` to download the .img file to the attached volume. For example:

     `curl <path_to_bootable_image> > <device>`
   - If the bootable image is from a source other than an http or FTP repository, copy the bootable image from its source to the helper instance, and then copy it to the attached volume using the `dd` command. For example:

     `dd if=<path_to_bootable_image> of=<device> bs=1M`
5. Detach the volume from the instance:

   `euca-detach-volume <volume-id>`
6. Create a snapshot of the volume:

   `euca-create-snapshot <volume-id>`
7. Register the snapshot:

   `euca-register --name <image-name> --snapshot <snapshot-id> --root-device-name <device>`

You've now created a EBS-backed image. To maintain data persistence, be sure to use `euca-stop-instances` and `euca-start-instances` to stop and start your EBS-backed instance.

Bundle an Image for Amazon EC2

EMIs can be uploaded unchanged to Amazon EC2 and run as AMIs in the public cloud. To upload an EMI image file to Amazon, do the following:

1. Locate the Amazon ec2 cert file that is provided as part of the EC2 AMI tools. This file is generally located in `$EC2_AMITOOL_HOME/etc/ec2/amitools/cert-ec2.pem`.
2. Enter the following command:

```
euca-bundle-image -i <image_name> -r <architecture>\
-c <cert_filename> -k <private_key_filename> \
--ec2cert <path_to_cert_file>
```

For example:

```
euca-bundle-image -i euca-centos-5.3 -r x86_64\
  x86_64/centos.5-3.x86-64.img  -u 123456789111 -c cert- \
  abc.pem -k pk-abc.pem --ec2cert \
  $EC2_AMITOOL_HOME/etc/ec2/amitools/cert-ec2.pem
```

Bundle a Windows Instance

The euca-bundle-instance command lets you bundle a new Windows image from a running Windows instance directly to Walrus storage. Using euca-bundle-instance is an efficient way to generate modified Windows VMs. You can spin up a Windows VM instance from an existing Windows VM image, modify it as needed, then save the modified image to Walrus storage, where it is immediately available for registering and running with the modifications already in place.

To bundle a Windows instance:

Enter the following command:

```
euca-bundle-instance -b <bucket_name> -p
<prefix_starting_with_windows> -o $EC2_ACCESS_KEY -w
$EC2_SECRET_KEY <instance_ID>
```

Tip: You can query the progress of a bundling tasks using the `euca-describe-bundle-task` command. You can cancel the active bundle task using `euca-cancel-bundle-task` command.

The following example bundles and registers a new Windows image from an existing Windows instance with ID `i-12c4af6a`, to the bucket `mybucket`, with image prefix `windows2003_bundle`:

```
euca-bundle-instance -b mybucket -p windows2003_bundle -o
$EC2_ACCESS_KEY -w $EC2_SECRET_KEY  i-12c4af6a

euca-register mybucket/windows2003_bundle.manifest.xml
```

Using Instances

After an virtual image is launched, the resulting running system is called an instance. This section gives information about instances and their basic characteristics. It also describes how to launch an instance and connect to it.

To find out more about what instances are, see *Instance Overview*.

To find out how to use CloudWatch, see *Instance Tasks*.

Instance Overview

An instance is a virtual machine. A virtual machine is essentially an operational private computer that contains an operating system, applications, network accessibility, and disk drives. Eucalyptus allows you to run instances from both Linux-based images and Windows-based images.

The following sections describe instance types in more detail.

Instance Concepts

This section describes conceptual information to help you understance instances.

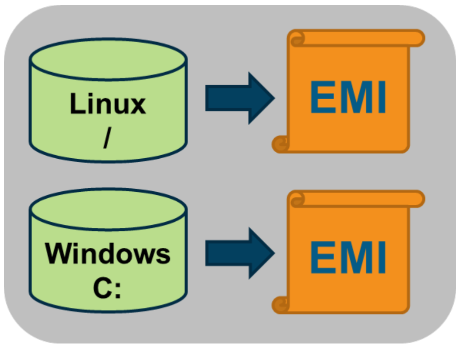
Eucalyptus Machine Image (EMI)

A Eucalyptus machine image (EMI) is a copy of a virtual machine bootable file system stored in the Walrus storage. An EMI is a template from you can use to deploy multiple identical instances—or copies of the virtual machine.

EMIs are analogous to Amazon Machine Images (AMIs) in AWS. In fact, you can download and deploy any of the 10,000+ AMIs as EMIs in a Eucalyptus cloud without significant modification. While it is possible to build your own EMI, it is might be just as simple to find a thoroughly vetted, freely available image in AWS, download it to your Eucalyptus cloud, and use that instead.

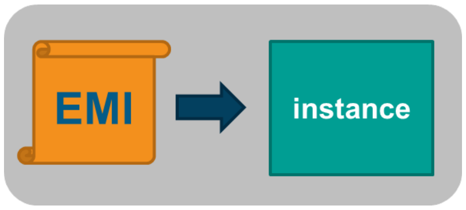
EMIs can be Linux or Windows-based. In Linux it is an image of the `/` file system while in Windows it is an image of the C: drive.

When registered in a Eucalyptus cloud, each distinct EMI is given a unique ID for identification. The ID is in the format emi-<nnnnnnnn>.



Instance

A instance is a virtual machine deployed from an EMI. An instance then, is simply a running copy of an EMI, which means it always starts from a known baseline. There are two types of instances; instance store-backed and EBS-backed. This section describes store-backed instances. For information about EBS-backed instances, see *Using EBS*.



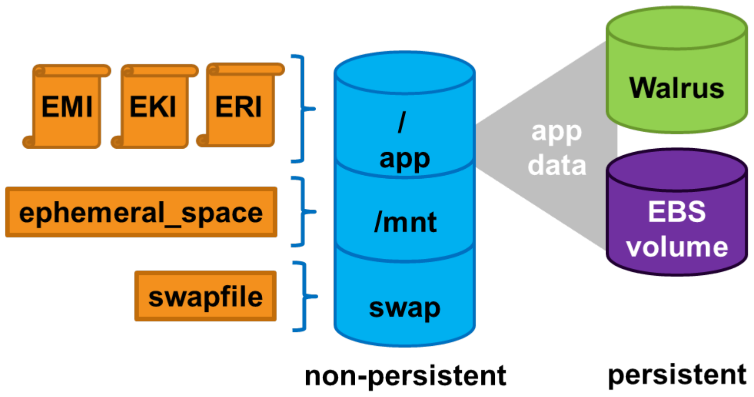
Every instance receives a unique ID in the format i-<nnnnnnnn>. In System and the two Managed network modes, the eight hexadecimal digits in the instance ID become the last four octets of the MAC address of the instance, prefaced by D0:0D. For example, if your instance ID is i-12121212, the MAC address of that instance would be D0:0D:12:12:12:12.

You can deploy multiple instances using a single command. In this case all the instances will have unique instance IDs but will share a common reservation ID. This reservation ID can be seen, for example, from the euca2ools `euca-describe-instances` command that lists running instances. Reservations IDs appear in the format r-<nnnnnnnn>.

Tip: A *reservation* is an EC2 term used to describe the resources reserved for one or more instances launched in the cloud.

Persistence

Applications running in ephemeral instances that generate data that must be saved should write that data to some place other than the instance itself. There are two Eucalyptus options available. First, the data can be written to a volume that is attached to the instance. Volumes provided by the Storage Controller and attached to instances are persistent. Second, the data could be written to the Walrus using HTTP put/get operations. Walrus storage is also persistent.



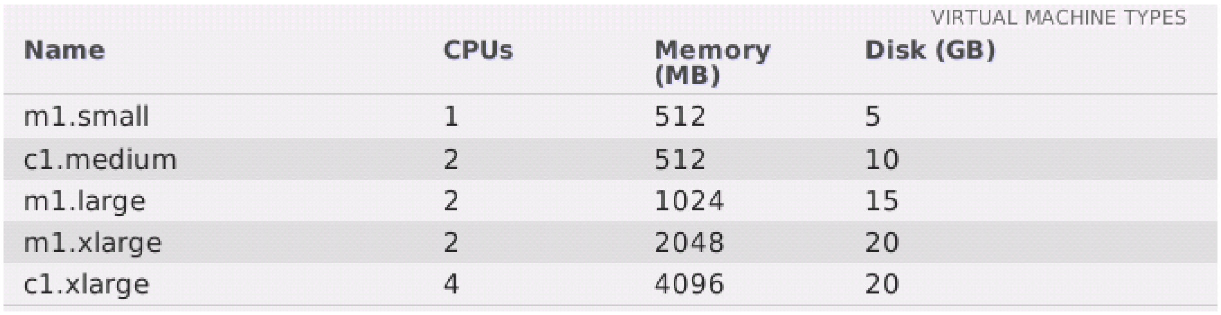
If the application cannot be rewritten to send data to a volume or the Walrus, then the application should be deployed inside an EBS-backed instance. EBS-backed instances are persistent and operate in a manner more similar to a physical machine.

Instance Basics

This section describes information to help you decide which type of instance you need.

Virtual Machine Types

A virtual machine type, known as a VM type, defines the number of CPUs, the size of memory, and the size of storage that is given to an instance when it boots. There are five pre-defined VM types in Eucalyptus. You can change the quantity of resources associated with each of the five VM types, but you cannot change the name of the VM types or the number of VM types available. If you customize the sizes they must be well-ordered. That means that the CPU, memory, and storage sizes of the next VM type must be equal to, or larger than, the size of the preceding VM type.



VIRTUAL MACHINE TYPES

Name	CPUs	Memory (MB)	Disk (GB)
m1.small	1	512	5
c1.medium	2	512	10
m1.large	2	1024	15
m1.xlarge	2	2048	20
c1.xlarge	4	4096	20

The VM type used to instantiate an EMI must have a defined disk size larger than the EMI file. If a 6GB EMI is loaded into an instance with a VM type defined with a 5GB disk, it will fail to boot. The status of the instance will show as *pending*. The pending status is the result of the fact that the Walrus cannot finish downloading the image to the Node Controller because the Node Controller has not allotted sufficient disk space for the download. Starting with Eucalyptus 3.2, if the user attempts to launch an instance with a VM type that is too small, they will receive an on-screen warning and the operation will terminate.

Available VM Types

Eucalyptus, like AWS, offers families of VM types. These families are composed of varying combinations of CPU, disk size, and memory. Eucalyptus offers enough VM types to give you the flexibility to choose the appropriate mix of resources for your applications. For the best experience, we recommend that you launch instance types that are appropriate for your applications.

- **General Purpose:** This family includes the M1 and M3 VM types. These types provide a balance of CPU, memory, and network resources, which makes them a good choice for many applications. The VM types in this family range in size from one virtual CPU with two GB of RAM to eight virtual CPUs with 30 GB of RAM. The balance of resources makes them ideal for running small and mid-size databases, more memory-hungry data processing tasks, caching fleets, and backend servers.

  M1 types offer smaller instance sizes with moderate CPU performance. M3 types offer larger number of virtual CPUs that provide higher performance. We recommend you use M3 instances if you need general-purpose instances with demanding CPU requirements.
- **Compute Optimized:** This family includes the C1 and CC2 instance types, and is geared towards applications that benefit from high compute power. Compute-optimized VM types have a higher ratio of virtual CPUs to memory than other families. We recommend this type if you are running any CPU-bound scale-out applications. CC2 instances provide high core count (32 virtual CPUs) and support for cluster networking. C1 instances are available in smaller sizes and are ideal for scaled-out applications at massive scale.
- **Memory Optimized:** This family includes the CR1 and M2 VM types and is designed for memory-intensive applications. We recommend these VM types for performance-sensitive database, where your application is memory-bound. CR1 VM types provide more memory and faster CPU than do M2 types. CR1 instances also support cluster networking for bandwidth intensive applications. M2 types are available in smaller sizes, and are an excellent option for many memory-bound applications.
- **Storage Optimized:** This family includes the HI1 and HS1 instance types, and provides you with direct-attached storage options optimized for applications with specific disk I/O and storage capacity requirements. HI1 instances are optimized for very high random I/O performance. These instances can deliver over 120,000 4k random read IOPS making them ideal for transactional applications. HS1 instances are optimized for very high storage density and high sequential I/O performance. We recommend HS1 VM types for large-scale data warehouses, large always-on Hadoop clusters, and for cluster file systems.
- **Graphics Processing Optimized:** This family includes the CG1 instance type, and allows you to take advantage of the parallel performance of NVidia Tesla GPUs using the CUDA or OpenCL programming models for GPGPU computing. This VM type also provides high CPU capabilities and supports cluster networking. We recommend this VM type for tasks that require parrallel processing of large blocks of data.

- **Micro:** This Micro family contains the T1 VM type. The t1.micro provides a small amount of consistent CPU resources and allows you to increase CPU capacity in short bursts when additional cycles are available. We recommend this type for lower throughput applications like a proxy server or administrative applications, or for low-traffic websites that occasionally require additional compute cycles. We do not recommend this VM type for applications that require sustained CPU performance.

The following tables list each VM type Eucalyptus offers. Each type is listed in its associate VM family.

Table 1: General Purpose VM Types

Instance Type	Virtual CPU	Disk Size	Memory
m1.small	1	5	256
m1.medium	1	10	512
m1.large	2	10	512
m1.xlarge	2	10	1024
m3.xlarge	4	15	2048
m3.2xlarge	4	30	4096

Table 2: Compute Optimized VM Types

Instance Type	Virtual Cores	Disk Size	Memory
c1.medium	2	10	512
c1.xlarge	2	10	2048
cc1.4xlarge	8	60	3072
cc2.8xlarge	16	120	6144

Table 3: Memory Optimized VM Types

Instance Type	Virtual Cores	Disk Size	Memory
m2.xlarge	2	10	2048
m2.2xlarge	2	30	4096
m2.4xlarge	8	60	4096
cr1.8xlarge	16	240	16384

Table 4: Storage Optimized VM Types

Instance Type	Virtual Cores	Disk Size	Memory
hi1.4xlarge	8	120	6144
hs1.8xlarge	48	24000	119808

Table 5: Micro VM Types

Instance Type	Virtual Cores	Disk Size	Memory
t1.micro	1	5	256

Table 6: Graphics Processing VM Types

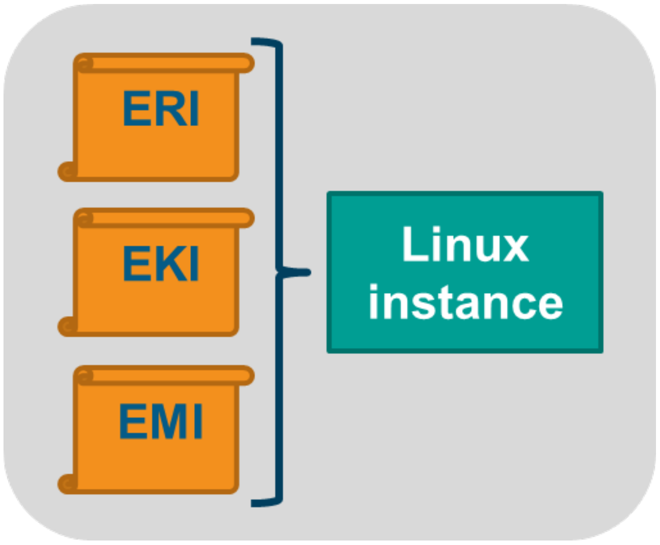
Instance Type	Virtual Cores	Disk Size	Memory
cg1.4xlarge	16	200	16384

Linux and Windows Instances

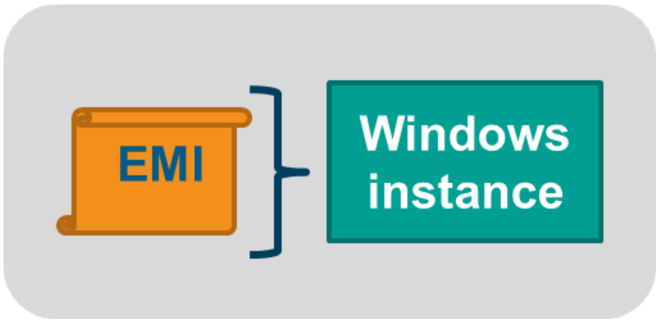
A Linux instance store-backed instance actually consists of three separate images:

- The Linux boot disk image (EMI) as previously defined
- A Eucalyptus kernel image (EKI), which is a low level operating system component that interfaces with the instance's virtual hardware
- A Eucalyptus ramdisk image (ERI) - the initrd - which is the initial root file system that is loaded as part of the kernel boot procedure and loads modules that make it possible to access the real root file system in the second stage of the boot process

When a Linux instance is launched, these three images (along with a few other files) are bound together using loop devices so that they appear as a single disk to the Linux operating system running in the instance. For Linux images, the same kernel and ramdisk files can be shared across multiple boot disk images, which allows for more efficient use of storage.



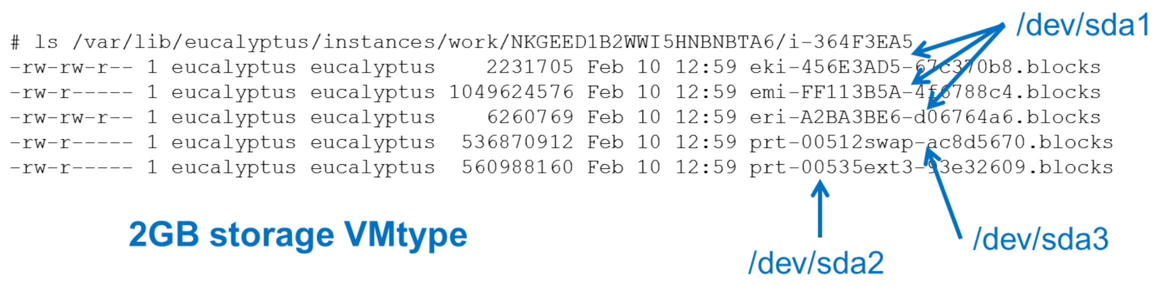
A Windows instance store-backed instance consists only of the bootable file system image (EMI). The reason for this is that in the Windows operating system the kernel and ramdisk components cannot be separated from the boot disk image, and thus each copy of a Windows image must also contain a copy of these components.



Ephemeral Linux Instances

Instance store-backed instances are ephemeral instances. This means that any changes made to a running instance are lost if the instance is either purposely or accidentally terminated. Applications running in ephemeral instances should write their data to persistent storage for safe keeping. Persistent storage available to instances includes Storage Controller volumes and the Walrus.

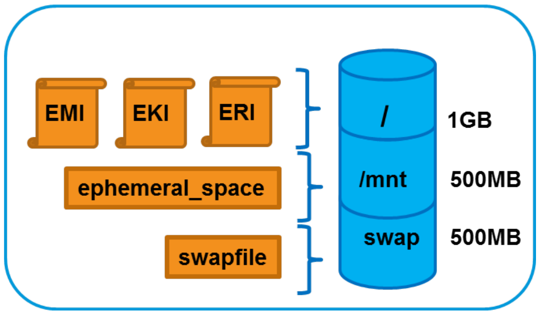
As an instance store-backed instance is launched, several files are brought together using loop devices on the Node Controller. As these files are brought together they form what looks like a disk to the instance's operating system. The illustration below lists a some of the files that make up a running instance. Notice that the EKI, EMI, and ERI images are presented to the instance's operating system as the partition `/dev/sda1` and are mounted as the `/` file system.



```
# ls /var/lib/eucalyptus/instances/work/NKGEED1B2WWI5HNBNBTA6/i-364F3EA5
-rw-rw-r-- 1 eucalyptus eucalyptus    2231705 Feb 10 12:59 eki-456E3AD5-67c370b8.blocks
-rw-r----- 1 eucalyptus eucalyptus 1049624576 Feb 10 12:59 emi-FF113B5A-4f6788c4.blocks
-rw-rw-r-- 1 eucalyptus eucalyptus    6260769 Feb 10 12:59 eri-A2BA3BE6-d06764a6.blocks
-rw-r----- 1 eucalyptus eucalyptus  536870912 Feb 10 12:59 prt-00512swap-ac8d5670.blocks
-rw-r----- 1 eucalyptus eucalyptus  560988160 Feb 10 12:59 prt-00535ext3-93e32609.blocks
```

Assume that the illustration above shows some of the files that make up an instance that was launched in a vmtype with 2GB of storage. Notice that the `eki-*`, `emi-*`, and `eri-*` files have been downloaded from the Walrus and cached on the Node Controller. These three files consume around 1.06GB of storage space. Notice also that a swap file was automatically created for the instance. The swap file has the string `swap` in its name and the file is approximately 500MB in size. It is presented to the instance's operating system as the partition `/dev/sda3`.

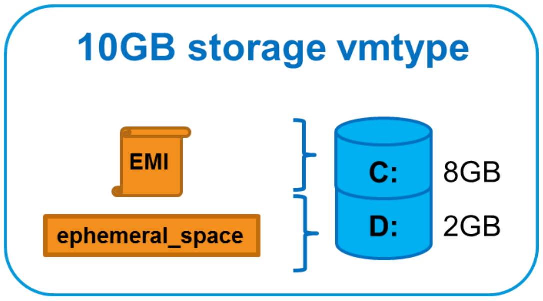
This means the EKI, ERI, EMI, and swap files have consumed approximately 1.5GB of the available 2GB of storage space. The remaining 500GB is allocated to the file with the string `ext3` in its name. In our example, this space is formatted as an ext3 file system and is made available to the instance as the disk partition `/dev/sda2`, and is actually mounted to the `/mnt` directory in the instance. An example of this configuration is shown below.



Ephemeral Windows Instances

It is also possible to launch ephemeral Windows instances. Just like their Linux counterparts, modifications to Windows instance store-backed instances are lost when instances are purposely or accidentally terminated.

Ephemeral space in a Windows instance store-backed instance is presented as formatted drive space accessible via a logical drive letter. The logical drive is sized so that the instance's total storage size matches the vmtype's storage size.



Windows Instance Support

Eucalyptus supports several different Windows operating system versions running as instances. Normal Windows licensing policies still apply. The Windows operating systems supported include:

- Windows Server 2003 R2 Enterprise (32/64-bit)
- Windows Server 2008 SP2 Datacenter (32/64-bit)
- Windows Server 2008 R2 Datacenter (32/64-bit)
- Windows 7 Professional (32/64-bit)

A VNC client is required during initial installation of the operating system but once Windows is fully installed, Remote Desktop Protocol can be used to connect to the Windows desktop. The VNC client is unnecessary if theWindow installation is configured as an unattended installation. As an example of one way to set up a Windows 7 unattended installation, see *http://www.intowindows.com/how-to-create-unattended-windows-7-installation-setup*.

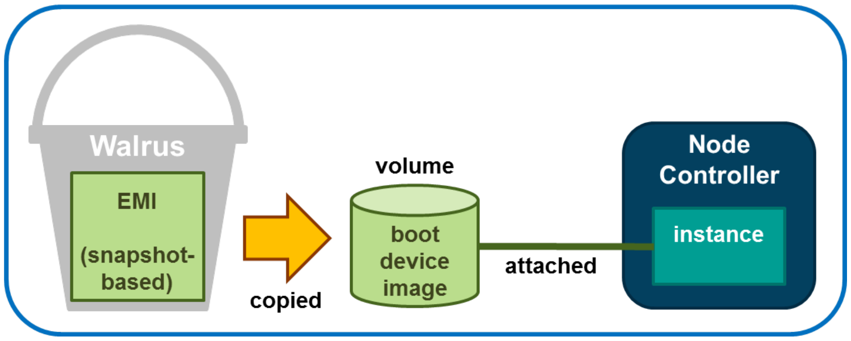
All Windows images should be created on the hypervisor that runs on the Node Controllers. Eucalyptus does not support running a Windows image across multiple hypervisors. The Eucalyptus Windows Integration software, installed in the Windows EMI, has a utility that adds permissions to users or groups that allows them to use RDP to access the Windows desktop. By default, only the Administrator can do this. The Eucalyptus Windows Integration software also installs the Virtio device drivers for disk and network into the EMI so that it can run on a host configured with a KVM hypervisor. For more information about how to create a Windows image and install the Eucalyptus Windows Integraion software, see the *Eucalyptus User Guide* at *http://www.eucalyptus.com/docs*.

EBS-Backed Instances

Eucalyptus supports two different types of instances; instance store-backed instances and EBS-backed instances. This section describes EBS-backed instances.

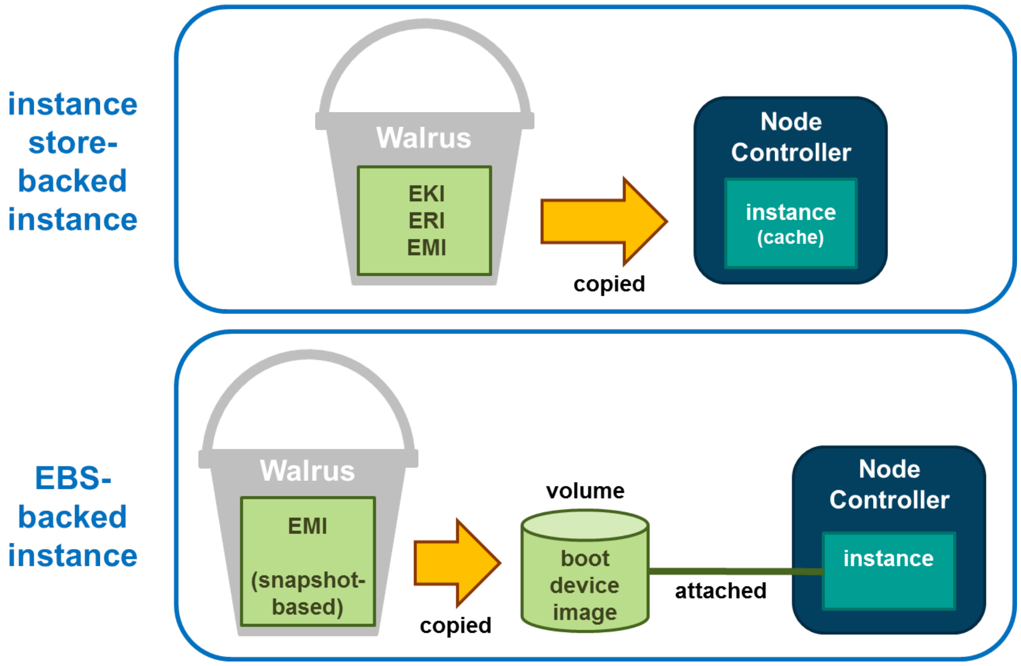
With EBS-backed instances you are booting an instance from a volume rather than a bundled EMI image. The boot volume is created from a snapshot of a root device volume. EBS-backed instances can be either Linux or Windows. The boot volume is persistent so changes to the instance are persistent.

Note: Linux boot-from-EBS instances do not require EKI and ERI images like instance store-backed instances.



Comparing Instance Types

The graphic below illustrates the differences between an instance store-backed instance and an EBS-backed instance. Both types of instances still boot from an EMI; the difference is what is behind the EMI. For an instance store-backed instance the EMI is backed by a bundled image. For an EBS-backed instance the EMI is backed by a snapshot of a volume that contains bootable software, similar to a physical host's boot disk.

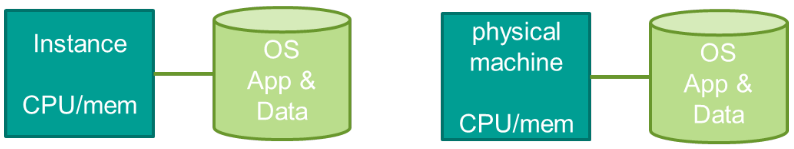


Note that for the instance store-backed instance, the EMI, EKI, and ERI (assuming Linux) are copied from the Walrus directly to Node Controller. Both disk and memory on the Node Controller are used as cache so an instance store-backed instance can be considered a cache-based instance and is not persistent. Once the instance is terminated both the RAM and the disk cache are cleared and any modifications to the instance are lost.

When an EBS-backed instance is launched, the snapshot on which the EMI is based is automatically copied to a new volume which is then attached to the instance. The instance then boots from the attached volume. Changes to the EBS-backed instance are persistent because the volume used to boot the EBS-backed instance is persistent. As a result, EBS-backed instances can be suspended and resumed and not just terminated like an instance store-backed instance.

Using EBS-Backed Instances

An EBS-backed instance is very much like a physical machine in the way it boots and persists data. Because an EBS-backed instance functions in a manner similar to physical machine, it makes it a good choice to run legacy applications that cannot be re-architected for the cloud.



EBS-backed instances provide a workaround for the 10GB size limit for instance store-backed EMI images. This is particularly important for Windows instances which can easily exceed 10GB in size. EBS-backed instances have a maximum size of 1TB.

An EBS-backed boot volume can still be protected by taking a snapshot of it. In fact, other non-boot volumes can be attached to the EBS-backed instance and they can be protected using snapshots too.

Suspending and Resuming EBS-Backed Instances

An EBS-backed instance can be suspended and resumed, similar to the operating system and applications on a laptop. The current state of the EBS-backed instance is saved in a suspend operation and restored in a resume operation. Like instance store-backed instances, an EBS-backed instance can also be rebooted and terminated.

To suspend a running EBS-backed instance:

```
euca-stop-instances i-<nnnnnnnn>
```

To resume a suspended EBS-backed instance:

```
euca-start-instances i-<nnnnnnnn>
```

To reboot an EBS-backed instance:

```
euca-reboot-instances i-<nnnnnnnn>
```

To terminate an EBS-backed instance:

```
euca-terminate-instances i-<nnnnnnnn>
```

EBS EMI Creation Overview

You can create an EBS EMI from an existing `.img` file or create your own `.img` file. One way to create your own EBS `.img` file is to use `virt-install` as described below.

Use `virt-install` on a system with the same operating system version and hypervisor as your Node Controller. When using `virt-install`, select *scsi* as the disk type for KVM if the VIRTIO paravirtualized devices are not enabled. If you have KVM with VIRTIO enabled (the default), select *virtio* as the disk type of the virtual machine. If you create, successfully boot, and connect the virtual machine to the network in this environment, it should boot as an EBS-backed instance in the Eucalyptus cloud.

Note: For CentOS or RHEL images, you will typically need to edit the `/etc/sysconfig/network-scripts/ifcfg-eth0` file and remove the `HWADDR` line. This is because an instance's network interface will always be assigned a different hardware address at instantiation.

Note: WARNING: If you use an image created by `virt-install` under a different distribution or hypervisor combination, it is likely that it will not install the correct drivers into the ramdisk and the image will not boot on your Node Controller.

To create an EMI for EBS-backed instances will require initial assistance from a *helper* instance. The helper instance can be either an instance store-backed or EBS-backed instance and can be deleted when finished. It only exists to help create the initial volume that will be the source of the snapshot behind the EMI used to boot other EBS-backed instances.

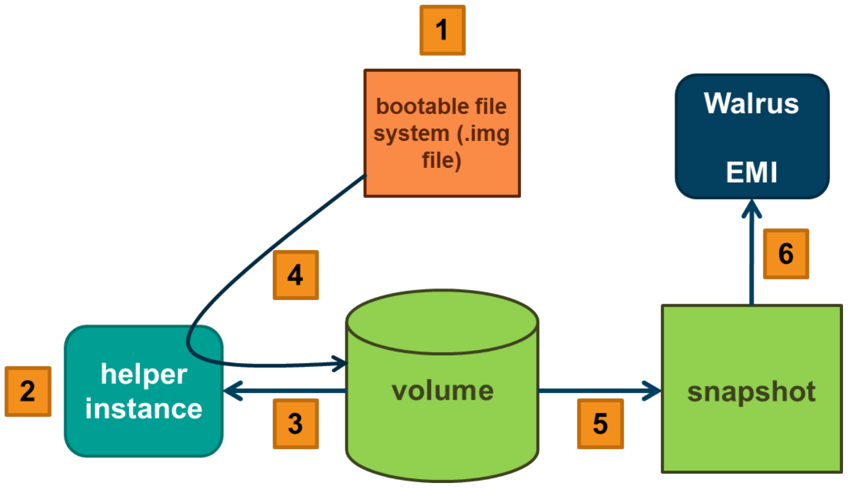
First you will need to create a volume large enough to contain the boot image file that was created by `virt-install`. Once this volume has been created attach it to the helper instance. Then transfer the boot image file to the helper instance. The helper instance must have enough free disk space to temporarily hold the boot image file. Once there, transfer the boot image file, using the `dd` command, to the attached volume.

At this point the volume can be detached from the helper instance, a snapshot taken, and the snapshot registered as a new EMI.

The process to create a new EMI is summarized as follows:

1. Create the `.img` file using `virt-install` (the `.img` file is the virtual machine's disk file).
2. Create the helper instance.
3. Create and attach the volume to the helper instance.
4. Copy the `.img` file to the helper instance and from there to the attached volume.
5. Detach the volume and take a snapshot of it.
6. Register the snapshot as a new EMI.

This process is illustrated below.



Create an EBS EMI

To create a new EMI that is used to boot EBS-backed instances:

1. Create a new volume whose size matches the size of the bootable `.img` file:

```
euca-create-volume -z <cluster_name> -s <size_GB>
```

2. Attach the volume to a helper instance:

```
euca-attach-volume vol-<nnnnnnnn> -i i-<nnnnnnnn> -d <device>
```

3. Log in to the instance and copy the bootable image from its source to the helper instance.
4. While logged in to the helper instance, copy a bootable image to the attached volume:

```
dd if=/<path_to_image> of=<device> bs=1M
```

5. While logged in to the helper instance, flush the file system buffers after running the `dd` command:

```
sync
```

6. Detach the volume from the instance:

```
euca-detach-volume vol-<nnnnnnnn>
```

7. Create a snapshot of the bootable volume:

```
euca-create-snapshot vol-<nnnnnnnn>
```

8. Register the bootable volume as a new EMI:

```
euca-register --name "<descriptive_name>" /
   --snapshot snap-<nnnnnnnn>
```

9. Run a new EBS-backed instance:

```
euca-run-instances -k <key_name> emi-<nnnnnnnn>
```

Note: The snapshot cannot be deleted unless the EMI is first deregistered.

Instance Tasks

This section describes the tasks you can perform with instances in Eucalyptus.

Find an Image

To find an image:

1. Enter the following command:

```
euca-describe-images
```

   The output displays all available images.

```
IMAGE emi-EC1410C1 centos-32/centos.5-3.x86.img.manifest.xml
admin available public  x86_64 machine
IMAGE eki-822C1344 kernel-i386/vmlinuz-2.6.28-11-server.manifest.xml
admin available public  x86_64 kernel
IMAGE eri-A98C13E4  initrd-64/initrd.img-2.6.28-11-generic.manifest.xml
admin available public  x86_64 ramdisk
```

2. Look for the image ID in the second column and write it down. The image ID starts with `emi-`.

Once you find a suitable image to use, make sure you have a *keypair* to use.

Create Key Pairs

Eucalyptus uses cryptographic key pairs to verify access to instances. Before you can run an instance, you must create a key pair. Creating a key pair generates two keys: a public key (saved within Eucalyptus) and a corresponding private key (output to the user as a character string). To enable this private key you must save it to a file and set appropriate access permissions (using the chmod command), as shown in the example below.

When you create a VM instance, the public key is then injected into the VM. Later, when attempting to login to the VM instance using SSH, the public key is checked against your private key to verify access. Note that the private key becomes obsolete when the public key is deleted.

Create Key Pairs with the Console

1. From the main dashboard screen, click the **Key Pairs** link in the **Network and Security** section, or select the Network and Security submenu from the Manage Resources navigation menu. The **Manage Keypairs** screen will appear.
2. On the **Manage Key Pairs** screen, click the **Create new key pair** link. The **Create New Key Pair** dialog will appear.
3. Type a name for the new key pair into the **Name** text box.
4. Click the **Create and Download** button. The private half of the key pair is saved to the default download location for your browser.

   Note: Keep your private key file in a safe place. If you lose it, you will be unable to access instances created with the key pair.

5. Change file permissions to enable access to the private key file in the local directory. For example, on a Linux or Mac OS X system:

```
chmod 0600 <keypair_name>.private
```

Create Key Pairs with the Command Line

1. Enter the following command:

```
euca-create-keypair <keypair_name> > <keypair_name>.private
```

   where `<keypair_name>` is a unique name for your keypair. For example:

```
euca-create-keypair alice-keypair > alice-keypair.private
```

   The private key is saved to to a file in your local directory.
2. Change file permissions to enable access to the private key file in the local directory:

```
chmod 0600 <keypair_name>.private
```

3. Query the system to view the public key:

```
euca-describe-keypairs
```

   The command returns output similar to the following:

```
KEYPAIR alice-keypair
ad:0d:fc:6a:00:a7:e7:b2:bc:67:8e:31:12:22:c1:8a:77:8c:f9:c4
```

Authorize Security Groups

Before you can log in to an instance, you must authorize access to that instance. This done by configuring a security group for that instance.

A security group is a set of networking rules applied to instances associated with a group. When you first create an instance, it is assigned to a default security group that denies incoming network traffic from all sources. To allow login and usage of a new instance, you must authorize network access to the default security group with the euca-authorize command.

To authorize a security group, use euca-authorize with the name of the security group, and the options of the network rules you want to apply.

```
euca-authorize <security_group>
```

Use the following command to grant unlimited network access using SSH (TCP, port 22) and VNC (TCP, ports 5900 to 5910) to the security group `default`:

```
euca-authorize -P tcp -p 22 -s 0.0.0.0/0 default
euca-authorize -P tcp -p 5900-5910 -s 0.0.0.0/0 default
```

Use the following command to grant unlimited network access using Windows Remote Desktop (TCP, port 3389) to the security group `windows`:

```
euca-authorize -P tcp -p 3389 -s 0.0.0.0/0 windows
```

Launch an Instance

To launch an instance:

1. Use the euca-run-instances command and provide an image ID and the name of a keypair, in the format `euca-run-instances <image_id> -k <mykey>` . For example:

   ```
   euca-run-instances emi-EC1410C1 -k alice-keypair
   ```

   Eucalyptus returns output similar to the following example.

   ```
   RESERVATION r-460007BE alice alice-default
   INSTANCE i-2F930625 emi-EC1410C1 0.0.0.0 0.0.0.0 pending alice-keypair
   2010-03-29T23:08:45.962Z eki-822C1344 eri-BFA91429
   ```

2. Enter the following command to get the launch status of the instance:

   ```
   euca-describe-instances <instance_id>
   ```

Log in to an Instance

For a Linux Instance

When you create an instance, Eucalyptus assigns the instance two IP addresses: a public IP address and a private IP address. The public IP address provides access to the instance from external network sources; the private IP address provides access to the instance from within the Eucalyptus cloud environment. Note that the two IP addresses may be the same depending on the current networking mode set by the administrator. For more information on Eucalyptus networking modes, see the Eucalyptus Administrator’s Guide.

To use an instance you must log into it via ssh using one of the IP addresses assigned to it. You can obtain the instance’s IP addresses using the euca-describe-instances query as shown in the following example.

To log into a VM instance:

1. Enter the following command to view the IP addresses of your instance:

   ```
   euca-describe-instances
   ```

   Eucalyptus returns output similar to the following:

   ```
   RESERVATION r-338206B5 alice default
   INSTANCE i-4DCF092C  emi-EC1410C1  192.168.7.24   10.17.0.130   running
   alice-keypair  0  m1.small  2010-03-15T21:57:45.134Z
   ```

   Note that the public IP address appears after the image name, with the private address immediately following.

2. Look for the instance ID in the second field and write it down. Use this ID to manipulate and terminate this instance.

3. Use SSH to log into the instance, using your private key and the external IP address. For example:

```
ssh -i alice-keypair.private root@192.168.7.24
```

You are now logged in to your Linux instance.

For a Windows Instance:

Log into Windows VM instances using a Remote Desktop Protocol (RDP) client. An RDP prompts you for a login name and password. By default, Windows VM instances are configured with a single user (named `Administrator`) and a random password generated at boot time. So, before you can log into a Windows VM instance via RDP, you must retrieve the random password generated at boot time using the `euca-get-password` command.

To log into a Windows instance:

1. Use `euca-describe-groups` to make sure the port for remote desktop (3389) is authorized in your security group.

   The response from this command will look like the following example.

```
GROUP 955340183797 default default group
PERMISSION 955340183797 default ALLOWS tcp 3389 3389 FROM CIDR 0.0.0.0/0
```

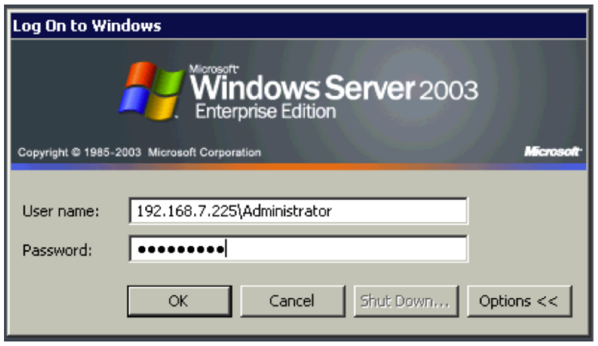
2. Enter the `euca-get-password` command followed by the unique id tag of the Windows VM instance and the `-k` option with the name of private key file that corresponds to your credential keypair. In the following example we retrieve the password for a Windows VM instance with id tag `i-5176095D` and private key file name `mykey.private`.

```
euca-get-password i-5176095D -k mykey.private
```

3. Log into the RDP client using the public (external) IP address associated with the running Windows VM instance. Enter the following command to view the IP addresses of your instance:

```
euca-describe-instances
```

4. At the **Log On to Windows** prompt, prepend the user name **Administrator** to the public IP address of the instance, and enter the password that you retrieved with `euca-get-password`, as shown:


Log On to Windows
Microsoft Windows Server 2003 Enterprise Edition
Copyright © 1985-2003 Microsoft Corporation
Microsoft
OK
Cancel
Shut Down...
Options <<

User name: 192.168.7.225\Administrator
Password: ●●●●●●●●●

You are now logged in and ready to use your Windows instance.

Reboot an Instance

Rebooting preserves the root filesystem of an instance across restarts. To reboot an instance:

Enter the following command:

```
euca-reboot-instances <instance_id>
```

To reboot the instance `i-34523332`, enter:

```
euca-reboot-instances i-34523332
```

Terminate an Instance

The euca-terminate-instances command lets you cancel running VM instances. When you terminate instances, you must specify the ID string of the instance(s) you wish to terminate. You can obtain the ID strings of your instances using the euca-describe-instances command.

Warning: Terminating an instance can cause the instance and all items associated with the instance (data, packages installed, etc.) to be lost. Be sure to save any important work or data to Walrus or EBS before terminating an instance.

To terminate VM instances:

1. Enter `euca-describe` instances to obtain the ID of the instances you wish to terminate. Note that an instance ID strings begin with the prefix `i-` followed by an 8-character string:

```
euca-describe-instances
RESERVATION r-338206B5 alice default
INSTANCE i-4DCF092C  emi-EC1410C1 192.168.7.24 10.17.0.130
running  mykey  0  m1.small  2010-03-15T21:57:45.134Z
wind  eki-822C1344  eri-BFA91429
```

2. Enter `euca-terminate-instances` and the ID string(s) of the instance(s) you wish to terminate:

```
euca-terminate-instances i-4DCF092C
 INSTANCE i-3ED007C8
```

Using EBS

This section details what you need to know about Eucalyptus Elastic Block Storage (EBS).

EBS Overview

Eucalyptus Elastic Block Storage (EBS) provides block-level storage volumes that you can attach to instances running in your Eucalyptus cloud. An EBS volume looks like any other block-level storage device when attached to a running Eucalyptus instance, and may be formatted with an appropriate file system and used as you would a regular storage device. Any changes that you make to an attached EBS volume will persist after the instance is terminated.

You can create an EBS volume by using the Eucalyptus command line tools or by using the Eucalyptus user console and specifying the desired size (up to 10GB) and availability zone. Once you've created an EBS volume, you can attach it to any instance running in the same availability zone. An EBS volume can only be attached to one running instance at a time, but you can attach multiple EBS volumes to a running instance.

You can create a backup — called a *snapshot* — of any Eucalyptus EBS volume. You can create a snapshot by using the command-line tools or the Eucalyptus user console and simply specifying the volume from which you want to create the snapshot, along with a description of the snapshot. Snapshots can be used to create new volumes.

EBS Tasks

This section contains examples of the most common Eucalyptus EBS tasks.

Create and Attach an EBS Volume

The following example shows how to create a 10 gigabyte EBS volume and attach it to a running instance called `i-00000000` running in availability zone `zone1`.

1. Create a new EBS volume in the same availability zone as the running instance.

   ```
   euca-create-volume --zone zone1 --size 10
   ```

   The command displays the ID of the newly-created volume.
2. Attach the newly-created volume to the instance, specifying the local device name `/dev/sdf`.

   ```
   euca-attach-volume vol-00000000 -i i-00000000 -d /dev/sdf
   ```

   You've created a new EBS volume and attached it to a running instance. You can now access the EBS volume from your running instance.

List Available EBS Volumes

You can use the Eucalyptus command line tools to list all available volumes and retrieve information about a specific volume.

1. To get a list of all available volumes in your Eucalyptus cloud, enter the following command:

   ```
   euca-describe-volumes
   ```

2. To get information about one specific volume, use the `euca-describe-volumes` command and specify the volume ID.

```
euca-describe-volumes vol-00000000
```

Detach an EBS Volume

To detach a block volume from an instance:

Enter the following command:

```
euca-detach-volume <volume_id>
```

```
euca-detach-volume vol-00000000
```

Create a Snapshot

The following example shows how to create a snapshot.

Enter the following command:

```
euca-create_snapshot <volume_id>
```

```
euca-create-snapshot vol-00000000
```

List Available Snapshots

You can use the Eucalyptus command line tools to list available snapshots and retrieve information about a specific snapshot.

1. To get a list of all available snapshots in your Eucalyptus cloud, enter the following command:

```
euca-describe-snapshots
```

2. To get information about one specific snapshot, use the `euca-describe-snapshots` command and specify the snapshot ID.

```
euca-describe-snapshots snap-00000000
```

Delete a Snapshot

The following example shows how to delete a snapshot.

Enter the following command:

```
euca-delete_snapshot <snapshot_id>
```

```
euca-delete-snapshot mytestsnaphot
```

Using Tags and Filters

This section describes features and tasks that enable you to manage your cloud resources, such as EMIs, instances, EBS volumes, and snapshots.

Tagging and Filtering Overview

This section describes what tagging is, its restrictions, and how tagging relates to filtering.

Tagging Resources

To help you manage your cloud's instances, images, and other Eucalyptus resources, you can assign your own metadata to each resource in the form of tags. You can use tags to create user-friendly names, make resource searching easier, and improve coordination between multiple users. This section describes tags and how to create them.

Tagging Overview

A tag consists of a key and an optional value for that key. You define both the key and the value for each tag. For example, you can define a tag for your instances that helps you track each instance's owner and stack level.

Tags let you categorize your cloud resources in various ways. For example, you can tag resources based on their purpose, their owner, or their environment. We recommend that you devise a set of tag keys that meets your needs for each resource type. Using a consistent set of tag keys makes it easier for you to manage your resources. You can search and filter the resources based on the tags you add. For more information about filtering, see *Filtering Resources*.

Eucalyptus doesn't apply any semantic meaning to your tags. Instead, Eucalyptus interprets your tags simply as strings of characters. Eucalyptus doesn't automatically assign any tags on resources.

You can only assign tags to resources that already exist. However, if you use the Eucalyptus User Console, you can add tags when you launch an instance. 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.

Tagging Restrictions

The following restrictions apply to tags:

Restriction	Description
Maximum number of tags per resource	10
Maximum key length	128 Unicode characters
Maximum value length	256 Unicode characters
Unavailable prefixes	You can't use either `euca:` or `aws:` as a prefix to either a tag name or value. These are reserved by Eucalyptus.
Case sensitivity	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. For example, to delete snapshots that you tagged with a tag key called `temporary`, you must first get a list of those snapshots using `euca-describe-snapshots` with a filter that specifies the tag. Then you use `euca-delete-snapshots` with the IDs of the snapshots (for example, snap-1A2B3C4D). You can't call `euca-delete-snapshots` with a filter that specified the tag. For more information about using filters when listing your resources, see *Filtering Resources*.

Available Resources

You can tag the following cloud resources:

- Images (EMIs, kernels, RAM disks)
- Instances
- Volumes
- Snapshots
- Security Groups

You can't tag the following cloud resources:

- Elastic IP addresses
- Key pairs
- Placement groups

You can tag public or shared resources, but the tags you assign are available only to your account and not to the other accounts sharing the resource.

Filtering Resources

You can search and filter resources based on the tags you use. For example, you can list a Eucalyptus Machine Image (EMI) using `euca-describe-images`, and you can list instances using `euca-describe-instances`.

Filtering Overview

Results from describe commands can be long. Use a filter to include only the resources that match certain criteria. For example, you can filter so that a returned list shows only the 64-bit Windows EMIs that use an EBS volume as the root device volumes.

Filtering also allows you to:

- Specify multiple filter values: For example, you can list all the instances whose type is either m1.small or m1.large.
- 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.
- Use wildcards with values: For example, you can use *production* as a filter value to get all EBS snapshots that include production in the description.

In each case, the instance must match all your filters to be included in the results. Filter values are case sensitive. We support only exact string matching, or substring matching (with wildcards).

Tagging and Filtering Tasks

This section details the tasks that you can with tagging and filtering. You can use the User Managment Console or the command line interface.

Eucalyptus supports three commands for tagging:

- euca-create-tags
- euca-describe-tags
- euca-delete-tags

Eucalyptus supports the following commands for filtering:

- euca-describe-addresses
- euca-describe-availability-zones
- euca-describe-bundle-tasks
- euca-describe-groups
- euca-describe-images

- euca-describe-instances
- euca-describe-keypairs
- euca-describe-regions
- euca-describe-snapshots
- euca-describe-tags
- euca-describe-volumes

For detailed information about these commands, see the *Euca2ools Reference Guide*.

Add a Tag

Use the `euca-create-tags` command to add a tag to a resource.

Enter the following command and parameters:

```
euca-create-tags [resource_id] --tag [tag_key]=[tag_value]
```

Eucalyptus returns a the resource identifier and its tag key and value.

The following example shows how to add a tag to an EMI. This tag key is `dataserver` and has no value.

```
euca-create-tags emi-1A2B3C4D --tag dataserver
TAG  emi-1a2b3c4d  image  dataserver
```

The following example shows how to add the same two tags to both an EMI and an instance. One tag key is dataserver, which has no value, an empty string. The other tag key is stack, which has a value of `Production`.

```
euca-create-tags emi-1A2B3C4D i-6F5D4E3A --tag dataserver --tag
stack=Production
TAG  emi-1A2B3C4D  image  dataserver
TAG  emi-1A2B3C4D  image  stack  Production
TAG  i-6F5D4E3A  image  dataserver
TAG  i-6F5D4E3A  image  stack  Production
```

List Tags

Use the `euca-describe-tags` command to list your tags and filter the results different ways.

Enter the following command and parameters:

```
euca-describe-tags --filter resource-type=[resource_type]" --filter
"key=[key_name]" --filter "value=[key_value]"
```

The following example describes all the tags belonging to your account.

```
euca-describe-tags
TAG  emi-1A2B3C4D  image  database_server
TAG  emi-1A2B3C4D  image  stack  Production
TAG  i-5F4E3D2A  instance  database_server
TAG  i-5F4E3D2A  instance  stack  Production
TAG  i-12345678  instance  webserver
TAG  i-12345678  instance  stack  Test
```

The following example describes the tags for the resource with ID emi-1A2B3C4D.

```
ec2-describe-tags --filter "resource-id=emi-1A2B3C4D"
TAG  emi-1A2B3C4D  image  database_server
TAG  emi-1A2B3C4D  image  stack  Production
```

The following example describes the tags for all your instances.

```
ec2-describe-tags --filter "resource-type=instance"
TAG  i-5F4E3D2A  instance  database_server
TAG  i-5F4E3D2A  instance  stack  Production
TAG  i-12345678  instance  webserver
TAG  i-12345678  instance  stack  Test
```

The following example describes the tags for all your instances that have a tag with the key webserver.

```
ec2-describe-tags --filter "resource-type=instance" --filter
"key=database_server"
TAG  i-5F4E3D2A  instance  database_server
```

The following example describes the tags for all your instances that have a tag with the key stack that has a value of either Test or Production.

```
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
```

The following example describes the tags for all your instances that have a tag with the key Purpose that has no value.

```
ec2-describe-tags --filter "resource-type=instance" --filter
"key=Purpose" --filter "value="
```

Change a Tag's Value

To change a tag's value:

Enter the following command and parameters:

```
euca-create-tags [resource]  --tag [key=value]
```

Eucalyptus returns a the resource identifier and its tag key and value.

The following example changes the value of the stack tag for one of your EMIs from prod to test:

```
euca-create-tags emi-1a2b3c4d  --tag stack=dev
TAG  emi-1a2b3c4d  image  stack  dev
```

Delete a Tag

To delete a tag:

Enter the following command and parameters:

```
euca-delete-tags resource_id [resource_id]" --tag
"key=[key_value]" --tag "value=[key_value]"
```

Tip: If you specify a value, the tag is deleted only if its value matches the one you specified. If you specify an empty string as the value, the tag is deleted only if the tag's value is an empty string.

Eucalyptus does not return a message.

The following example deletes two tags assigned to an EMI and an instance:

```
euca-delete-tags emi-1A2B3C4D i-6F5D4E3A --tag appserver --tag
stack
```

The following example deletes a tag only if the value is an empty string:

```
euca-delete-tags snap-1A2B3C4D --tag Owner=
```

Filter by Tag

You can describe your resources and filter the results based on the tags. To filter by tag:

Enter the following command and parameter:

```
euca-describe-tags --filter resource-type=[resource_type]
```

The following example describes all your tags.

```
euca-describe-tags
TAG  emi-1A2B3C4D  image  appserver
TAG  emi-1A2B3C4D  image  stack  dev
TAG  i-5F4E3D2A  instance  appserver
TAG  i-5F4E3D2A  instance  stack  dev
TAG  i-12345678  instance  database_server
TAG  i-12345678  instance  stack  test
```

The following example describes the tags for a resource with ID emi-1A2B3C4D.

```
euca-describe-tags --filter "resource-id=emi-1A2B3C4D"
TAG  emi-1A2B3C4D  image  appserver
TAG  emi-1A2B3C4D  image  stack  dev
```

The following example describes the tags for all your instances.

```
euca-describe-tags --filter "resource-type=instance"
TAG  i-5F4E3D2A  instance  appserver
TAG  i-5F4E3D2A  instance  stack  dev
TAG  i-12345678  instance  database_server
TAG  i-12345678  instance  stack  test
```

Managing Access

Eucalyptus manages access to the cloud by policies attached to accounts, groups, and users. This section details access-related tasks you can perform once your administrator allows you access to Eucalyptus. These tasks are split into the following areas: tasks for groups, and tasks for users, and tasks for credential management.

Groups

Groups are used to share resource access authorizations among a set of users within an account. Users can belong to multiple groups.

Important: A group in the context of access is not the same as a security group.

This section details tasks that can be performed on groups.

Create a Group

Using the CLI

To create a group using the CLI:

Enter the following command:

```
euare-groupcreate -g <group_name>
```

Eucalyptus does not return anything.

Using the Eucalyptus Administrator Console

To create a group using the Eucalyptus Administrator Console:

1. Click **Accounts** in the Quick Links section.
   The **Accounts** page displays.
2. Click the **ID** of the account you want to add a group to.
   The account, name, and Registration status are highlighted.
3. Click **New groups** in the **Accounts** page.
   The **Create new groups** popup displays.
4. Enter the group name in the **Group name** field.

   Tip: You can add more than one group at a time. Every group you add, however, will be in the same path.
5. Enter the group path in the **Group path** field.
6. Click **OK**.

The group is associated with the account you chose. You can see the information if you select the account in the **Accounts** page and click the **Member groups** link, located in the **Properties** section of the screen.

Add a Group Policy

Using the CLI

To add a policy to a group using the CLI:

Enter the following command:

```
euare-groupaddpolicy -g <group_name> -p <policy_name> -e <effect> -a
       <actions> -o
```

The optional `-o` parameter tells Eucalyptus to return the JSON policy, as in this example:

```
{"Version":"2008-10-17","Statement":[{"Effect":"Allow",
"Action":["ec2:RunInstances"], "Resource":["*"]}]}
```

Using the Eucalyptus Administrator Console

To add a policy to a group using the Eucalyptus Administrator Console:

1. Click **Groups** in the Quick Links section.
   The **Groups** page displays.
2. Click the **ID** of the group you want to add a policy to.
   The ID, Name, Path, and Owner account line is highlighted.
3. Click **Add policy**.
   The **Add new policy** popup displays.
4. Enter the policy name in the **Policy name** field.
5. Enter the policy content in the **Policy content** field.
6. Click **OK**.

The policy is now added to the group.

Modify a Group

Modifying a group is similar to a "move" operation. Whoever wants to modify the group must have permission to do it on both sides of the move. That is, you need permission to remove the group from its current path or name, and put that group in the new path or name.

For example, if a group changes from one area in a company to another, you can change the group's path from `/area_abc/` to `/area_efg/`. You need permission to remove the group from `/area_abc/`. You also need permission to put the group into `/area_efg/`. This means you need permission to call `UpdateGroup` on both `arn:aws:iam::123456789012:group/area_abc/*` and `arn:aws:iam::123456789012:group/area_efg/*`.

Using the CLI

To modify a group using the CLI:

1. Enter the following command to modify the group's name:

   ```
   euare-groupmod -g <group_name> --new-group-name <new_name>
   ```

   Eucalyptus does not return a message.
2. Enter the following command to modify a group's path:

   ```
   euare-groupmod -g <group_name> -p <new_path>
   ```

   Eucalyptus does not return a message.

Using the Eucalyptus Administrator Console

To modify a group using the Eucalyptus Administrator Console:

1. Click **Groups** in the Quick Links section.
   The **Groups** page displays.
2. Click the **ID** of the group you want to rename.
   The group's **Properties** area displays.
3. In the **Name** field, enter the new name of the group.
4. In the **Path** field, enter the new path for the group.
5. Click **Save**.

The new group name displays in the **Groups** page.

Add a User to a Group

Using the CLI

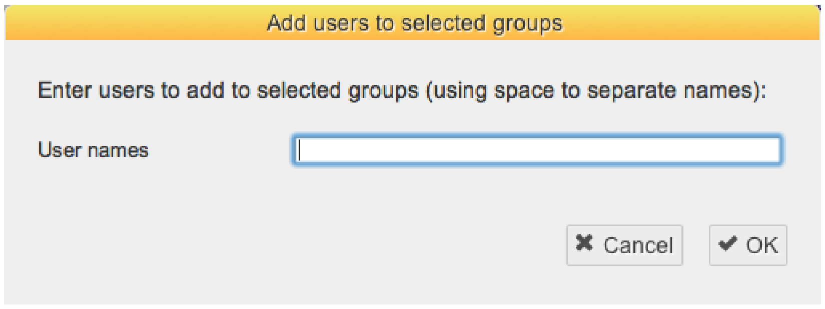
To add a user to a group using the CLI:

Enter the following command:

```
euare-groupadduser -g <group_name> -u <user-name>
```

Using the Eucalyptus Administrator Console

1. Click **Groups** in the Quick Links section.
   The **Groups** page displays.
2. Click the **ID** of the group you want to add the user to.
   The ID, Name, Path, and Owner account line is highlighted.
3. Click **Add users**.
   The **Add users to selected groups** popup displays.


Add users to selected groups

Enter users to add to selected groups (using space to separate names):

User names

4. Enter the name of the user you want to add and click **OK**.

The user is now added to the group.

Remove a User from a Group

Using the CLI

To remove a user from a group using the CLI:

Enter the following command:

```
euare-groupremoveuser -g <group_name> -u <user-name>
```

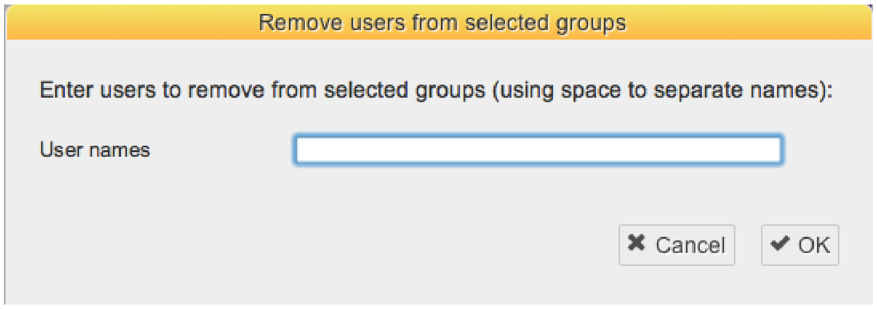
Using the Eucalyptus Administrator Console

To remove a user from a group using the Eucalyptus Administrator Console:

1. Click **Groups** in the Quick Links section.

The **Groups** page displays.

2. Click the **ID** of the group you want to remove the user from.
   The ID, Name, Path, and Owner account line is highlighted.
3. Click **Remove users**.
   The **Remove users to selected groups** popup displays.


Remove users from selected groups

Enter users to remove from selected groups (using space to separate names):

User names

4. Enter the name of the user you want to remove and click **OK**.

The user is now removed from the group.

Delete a Group

Using the CLI

When you delete a group, you have to remove users from the group and delete any policies from the group. You can do this with one command, using the `euare-groupdel` command with the `-r` option. Or you can follow the following steps to specify who and what you want to delete.

1. Individually remove all users from the group.

```
euare-groupremoveuser -g <group_name> -u <user_name>
```

2. Delete the policies attached to the group.

```
euare-groupdelpolicy -g <group_name> -p <policy_name>
```

3. Delete the group.

```
euare-groupdel -g <group_name>
```

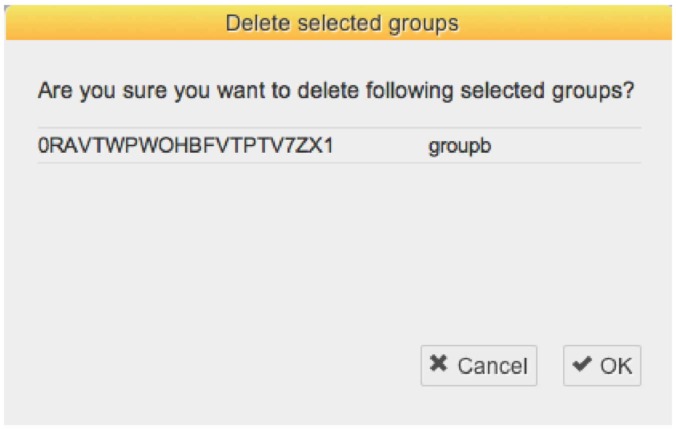
The group is now deleted.

Using the Eucalyptus Administrator Console

To delete a group using the Eucalyptus Administrator Console:

1. Click **Groups** in the Quick Links section.
   The **Groups** page displays.
2. Click the **ID** of the group you want to delete.
   The ID, Name, Path, and Owner account line is highlighted.
3. Click **Delete groups**.

   The **Delete selected groups** popup displays.


Delete selected groups

Are you sure you want to delete following selected groups?

0RAVTWPWOHBFVTPTV7ZX1	groupb

4. Click **OK**.

The group is now deleted.

List Users

You can list users within a path.

Using the CLI

Use the `euare-userlistbypath` command to list all the users in an account or to list all the users with a particular path prefix. The output lists the ARN for each resulting user.

```
euare-userlistbypath -p <path>
```

Using the Eucalyptus Administrator Console

To list users in the same path using the Eucalyptus Administrator Console:

1. Click **Users** in the Quick Links section.
   The **Users** page displays.
2. Click the **Path** column to sort all users by path.

Users

Users are subsets of accounts and are added to accounts by an appropriately credentialed administrator. While the term **user** typically refers to a specific person, in Eucalyptus, a **user** is defined by a specific set of credentials generated to enable access to a given account. Each set of user credentials is valid for accessing only the account for which they were created. Thus a user only has access to one account within a Eucalyptus system. If an individual person wishes to have access to more than one account within a Eucalyptus system, a separate set of credentials must be generated (in effect a new ‘user’) for each account (though the same username and password can be used for different accounts).

When you need to add a new user to your Eucalyptus cloud, you'll go through the following process:

1	*Create a user*
2	*Add user to a group*
3	*Give user a login profile*

Add a User

Using the CLI

To add a user using the CLI:

Enter the following command

```
euare-usercreate -u <user_name> -g <group_name> -k
```

Eucalyptus does not return a response.

Tip: If you include the `-v` parameter, Eucalyptus returns a response that includes the user's ARN and GUID.

Using the Eucalyptus Administrator Console

To add a user using the Eucalyptus Administrator Console:

1. Click **Accounts** in the Quick Links section.
   The **Accounts** page displays.
2. Click the **ID** of the account you want to rename.
   The account's **Properties** area displays.
3. Click **New Users**.
   The **Create new users** popup window displays.
4. Enter a name in the **User names** field.

   Tip: You can add more than one user at a time. Every user you add, however, will be in the same path.
5. Enter a path in the **User path** field.
6. Click **OK**.

The user is added to the account.

Create a Login Profile

Once you create a user, you must generate a password for the user to use the Eucalyptus Administrator Console.

Using the CLI

To create a login profile using the CLI:

Enter the following command:

```
euare-useraddloginprofile -u <user_name> -p <password>
```

Eucalyptus does not return a response.

Using the Eucalyptus Administrator Console

To create a login profile using the Eucalyptus Administrator Console:

1. Click **Users** in the Quick Links section.
   The **Users** page displays.
2. Click the **ID** of the user whose path you want to change.
   The user's **Properties** area displays.
3. Click **Password**.
   The **Change password** popup window displays.
4. Enter the new password in the **New user password** field, and repeat in the **New password again** field.
5. Click **OK**.

The login profile is now complete. If you are generating the password for a different user, let the user know the password and the URL to the Eucalyptus Administrator Console.

Modify a User

Modifying a user is similar to a "move" operation. Whoever wants to modify a user must have permission to do it on both sides of the move. That is, you need permission to remove the user from the current path or name, and put that user in the new path or name.

For example, if a user changes from one team in a company to another, you can change the user's path from `/team_abc/` to `/team_efg/`. You need permission to remove the user from `/team_abc/`. You also need permission to put the user into `/team_efg/`. This means you need permission to call UpdateUser on both `arn:aws:iam::123456789012:user/team_abc/*` and `arn:aws:iam::123456789012:user/team_efg/*`.

Using the CLI

To rename a user using the CLI:

1. Enter the following command to rename a user:

   ```
   euare-usermod -u <user_name> --new-user-name <new_name>
   ```

   Eucalyptus does not return a message.
2. Enter the following command:

   ```
   euare-groupmod -u <user_name> -p <new_path>
   ```

   Eucalyptus does not return a message.

Using the Eucalyptus Administrator Console

To rename a user using the Eucalyptus Administrator Console:

1. Click **Users** in the Quick Links section.
   The **Users** page displays.
2. Click the **ID** of the user you want to rename.
   The user's **Properties** area displays.
3. In the **Name** field, enter the new name of the user.
4. Click **Save**.

The new user name displays in the **Users** page.

Change User Path

Using the CLI

To change a user's path using the CLI:

Enter the following command:

```
euare-groupmod -u <user_name> -p <new_path>
```

Eucalyptus does not return a message.

Using the Eucalyptus Administrator Console

To change a group's path using the Eucalyptus Administrator Console:

1. Click **Users** in the Quick Links section.
   The **Users** page displays.
2. Click the **ID** of the user whose path you want to change.
   The user's **Properties** area displays.

3. In the **Path** field, enter the new path for the user.
4. Click **Save**.

The user's path is now changed and displays in the **Users** page.

Change User Password

Using the CLI

To change a user's password using the CLI:

Enter the following command:

```
euare-usermodloginprofile -u [username] -p [password]
```

Eucalyptus does not return a message.

Using the Eucalyptus Administrator Console

To change a user's password using the Eucalyptus Administrator Console:

1. Click **Users** in the Quick Links section.
   The **Users** page displays.
2. Click the **ID** of the user whose path you want to change.
   The user's **Properties** area displays.
3. Click **Password**.
   The **Change password** popup window displays.
4. Enter the new password in the **New user password** field, and repeat in the **New password again** field.
5. Click **OK**.

The user should now be able to log in using the new password.

List Users

You can list users within a path.

Using the CLI

Use the `euare-userlistbypath` command to list all the users in an account or to list all the users with a particular path prefix. The output lists the ARN for each resulting user.

```
euare-userlistbypath -p <path>
```

Using the Eucalyptus Administrator Console

To list users in the same path using the Eucalyptus Administrator Console:

1. Click **Users** in the Quick Links section.
   The **Users** page displays.
2. Click the **Path** column to sort all users by path.

List Groups

Using the CLI

To list all the groups a specific user is in:

Enter the following command:

```
euare-grouplistbypath
```

Eucalyptus returns a list of paths followed by the ARNs for the groups in each path. For example:

```
arn:aws:iam::eucalyptus:group/groupa
```

Using the Eucalyptus Administrator Console

To list groups using the Eucalyptus Administrator Console:

Click **Groups** in the Quick Links section.
The **Groups** page displays.

The **Groups** page displays all groups in your cloud.

Delete a User

Using the CLI

To delete a user using the CLI:

Enter the following command

```
euare-userdel -u <user_name>
```

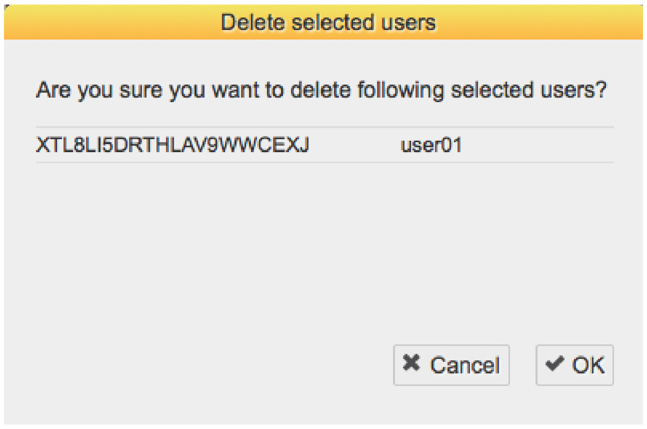
Eucalyptus does not return a response.

Using the Eucalyptus Administrator Console

To delete a user using the Eucalyptus Administrator Console:

1. Click **Users** in the Quick Links section.
   The **Users** page displays.
2. Click the **ID** of the user you want to delete.
   The user's information is highlighted.
3. Click **Delete Users**.

   The **Delete selected users** popup window displays.



Delete selected users

Are you sure you want to delete following selected users?

XTL8LI5DRTHLAV9WWCEXJ	user01

4. Click **OK**.

The user is deleted.

Credentials

Eucalyptus uses different types of credentials for different purposes. This section details tasks needed to allow access to Eucalyptus services.

Create Credentials

You can generate new credentials a number of ways. The first time you get credentials using either the Eucalyptus Administrator Console or the `euca_conf` command, a new secret access key is generated. On each subsequent request to get credentials, an existing active secret Key is returned. You can also generate new keys using the `eucare-useraddkey` command.

Tip: You can request to get a user's credentials either via the download link in the Eucalyptus Administrator Console or using euca_conf. Each request to get a user's credentials generates a new pair of a private key and X.509 certificate.

- To generate a new key for a user by an account administrator, enter the following

```
euare-useraddkey -u <user_name>
```

- To generate a private key and an X.509 certificate pair, enter the following:

```
euare-usercreatecert -u <user_name>
```

Get Credentials

Eucalyptus provides two main ways of getting user credentials. In both cases, Eucalyptus returns a zip file that contains keys, certificates, a bash script, and several other required files. To use these credentials with such CLI tools as euca2ools or ec2-tools, unzip your credentials zip file to a directory of your choice.

- An administrator with a root access to the machine on which CLC is installed can get credentials using euca_conf CLI tool on that machine.

```
/usr/sbin/euca_conf --cred-account <account> --cred-user <user_name>
 --get-credentials <filename>.zip
```

  Where <account> and <user_name> are the names of the account and the user whose credentials are retrieved.

  Tip: You can omit the `--cred-account` and `--cred-user` options when you get credentials for the **admin** user of the **eucalyptus** account.

- A user can get his or her credentials by logging in into the Eucalyptus Administrator Console and clicking **Download new credentials** in the drop-down menu at the top of the screen. This will result in a download of a zip file.

In the following example we download the credentials zip file to `~/.euca`, then change access permissions, as shown:

```
mkdir ~/.euca
cd ~/.euca
unzip <filepath>/<creds_zipfile>.zip
chmod 0700 ~/.euca
chmod 0600 *
```

Important: The zip file with credentials contains security-sensitive information. We recommend that you remove or read- and write-protect the file from other users after unzipping.

Alternatively, you can view and copy your access keys and X.509 certificates from the Eucalyptus Administrator Console after logging in, using the Navigation menu.

Upload a Certificate

To upload a certificate provided by a user:

Enter the following command:

```
euare-useraddcert -u <user_name> -f <cert_file>
```

Using VM Networking and Security

Eucalyptus provides networking modes that administrators can configure according to the network and security needs of the enterprise. Depending on the current networking mode configuration, users may have access to such features as elastic IPs, which are public (external) IP addresses that users can reserve and dynamically associate with VM instance; and security groups, which are sets of firewall rules applied to VM instances associated with the group. Euca2ools provides a means for users to interact with these features with commands for allocating and associating IP addresses, as well as creating, deleting, and modifying security groups.

Associate an IP Address with an Instance

To associate an IP address with an instance:

1. Allocate an IP address:

```
euca-allocate-address ADDRESS <IP_address>
```

2. Associate the allocated IP address with an instance ID:

```
euca-associate-address -i <instance_ID> <IP_address>
```

```
euca-associate-address -i i-56785678 192.168.17.103
```

Release an IP Address

Use euca-disassociate-address and euca-release-address to disassociate an IP address from an instance and to release the IP address to the global pool, respectively.

To release an IP address:

1. Enter the following command to disassociate an IP address from an instance:

```
euca-disassociate-address <IP_address>
```

2. Enter the following command to release an IP address:

```
euca-disassociate-address <IP_address>
```

The following example releases the IP address, `192.168.17.103`

```
euca-release-address 192.168.17.103
```

Create a Security Group

Security groups let you control network access to instances by applying network rules to instances associated with a group.

To create a security group:

Enter the following command:

```
euca-add-group -d <description> <group_name>
```

Tip: You can also create a security group you run an instance. Use the `euca-run-instances` command with the `-g` option. Security group rules only apply to incoming traffic thus all outbound traffic is permitted.

The following example creates a new security group named `mygroup` and described as `newgroup`.

```
euca-add-group -d "newgroup" mygroup
```

Delete a Security Group

The euca-delete-group command lets you delete security groups. To delete a security group:

Enter the following command:

```
euca-delete-group <group_name>
```

The following example deletes the security group, `mygroup`.

```
euca-delete-group mygroup
```

Authorize Security Group Rules

By default, a security group prevents incoming network traffic from all sources. You can modify network rules and allow incoming traffic to security groups from specified sources using the euca-authorize command.

To authorize security group rules:

Enter the following command:

```
euca-authorize -P <protocol> -p <port_number> \
-s <CIDR_source_network> <group_name>
```

The following example allows all incoming SSH traffic on port 22 to access to the security group `mygroup`. The CIDR source network, `0.0.0.0/0`, refers to any source.

```
euca-authorize -P tcp -p 22 -s 0.0.0.0/0 mygroup
 GROUP mygroup
 PERMISSION mygroup ALLOWS tcp 22 22 FROM CIDR
```

Instead of specifying a CIDR source, you can specify another security group. The following example allows access to the security group `mygroup` from the `someothergroup` security group using SSH on port 22.

```
euca-authorize --source-group someothergroup \
--source-group-user someotheruser -P tcp -p 22 mygroup
```

Revoke Security Group Rules

To revoke security group rules:

Enter the following command:

```
euca-revoke -P <protocol> -p <port_number> -s <CIDR_source_network>
<group_name>
```

The following example revokes the network rules authorized for the security group `mygroup`.

```
euca-revoke -P tcp -p 22 -s 0.0.0.0/0 mygroup
```

Using Auto Scaling

Eucalyptus Auto Scaling automatically adds and removes instances based on demand. Auto Scaling scales dynamically based on metrics (for example, CPU utilization). The Auto Scaling service works in conjunction with the ElasticLoad Balancing and CloudWatch services.

This chapter contains the following sections:

- *How Auto Scaling Works*
- *Auto Scaling Concepts*
- *Auto Scaling Usage Examples*

How Auto Scaling Works

Eucalyptus Auto Scaling is designed to address a common web application scenario: that you are running multiple copies of an application across several identical instances to adequately handle a certain volume of user requests. Eucalyptus Auto Scaling can help you make more efficient use of the computing resources in your cloud by automatically launching and terminating these instances, based on metrics and/or a schedule that you can define. There are three main Auto Scaling components that work together to provide this functionality: the *Auto Scaling group*, the *launch configuration*, and the *scaling plan*.

The *Auto Scaling group* contains the information about the instances that will be actually be used for scaling operations. The Auto Scaling group defines the minimum, desired, and maximum number of instances that you will use to scale your application.

The *launch configuration* contains all of the information necessary for Auto Scaling to launch instances, including instance type, image ID, key pairs, security groups, and block device mappings.

The *scaling policy* defines how to perform scaling actions. Scaling policies can execute automatically in response to CloudWatch alarms, or they you can execute them manually.

In addition to performing scaling operations based on the criteria defined in the scaling plan, Auto Scaling will periodically perform health checks to ensure that the instances in your Auto Scaling group are up and running. If an instance is determined to be unhealthy, Auto Scaling will terminate that instance and launch a new instance in order to maintain the minimum or desired number of instances in the scaling group.

Auto Scaling Concepts

This section discusses Auto Scaling concepts and terminology, and contains the following topics:

- *Understanding Launch Configurations*
- *Understanding Auto Scaling Groups*
- *Understanding Auto Scaling Policies*
- *Understanding Health Checks*
- *Understanding Instance Termination Policies*

Understanding Launch Configurations

The launch configuration defines the settings used by the Eucalyptus instances launched within an Auto Scaling group. This includes the image name, the instance type, key pairs, security groups, and block device mappings. You associate the launch configuration with an Auto Scaling group. Each Auto Scaling group can have one (and only one) associated launch configuration.

Once you've created a launch configuration, you can't change it; you must create a new launch configuration and then associate it with an Auto Scaling group. After you've created and attached a new launch configuration to an Auto Scaling group, any new instances will be launched using parameters defined in the new launch configuration; existing instances in the Auto Scaling group are not affected.

For more information, see *Creating a Basic Auto Scaling Configuration*.

Understanding Auto Scaling Groups

An Auto Scaling group is the central component of Auto Scaling. An Auto Scaling group defines the parameters for the Eucalyptus instances are used for scaling, as well as the minimum, maximum, and (optionally) the desired number of instances to use for Auto Scaling your application. If you don't specify the desired number of instances in your Auto Scaling group, the default value will be the same as the minimum number of instances defined.

In addition to instance and capacity definitions, each Auto Scaling group must specify one (and only one) launch configuration.

For more information, see *Creating a Basic Auto Scaling Configuration*.

Understanding Auto Scaling Policies

An Auto Scaling policy defines how to perform scaling actions in response to CloudWatch alarms. Auto scaling policies can either *scale-in*, which terminates instances in your Auto Scaling group, or *scale-out*, which will launch new instances in your Auto Scaling group. You can define an Auto Scaling policy based on demand, or based on a fixed schedule.

Demand-Based Auto Scaling Policies

Demand-based Auto Scaling policies scale your application dynamically based on CloudWatch metrics (such as average CPU utilization) gathered from the instances running in your scaling group. For example, you can configure a CloudWatch alarm to monitor the CPU usages of the instances in your Auto Scaling group. A CloudWatch alarm definition includes *thresholds* - minimum and maximum values for the defined metric - that will cause the alarm to fire.

For example, you can define the lower and upper thresholds of the CloudWatch alarm at 40% and 80% CPU usage. Once you've created the CloudWatch alarm, you can create a scale-out policy that launches 10 new instances when the CloudWatch alarm breaches the upper threshold (the average CPU usage is at or above 80%), and a scale-in policy that terminates 10 instances when the CloudWatch alarm breaches the lower threshold (the average CPU usage of the instances in the Auto Scaling group falls below 40%). Your Auto Scaling group will execute the appropate Auto Scaling policy when it receives the message from the CloudWatch alarm.

Note: For more information on CloudWatch, go to *Using CloudWatch*.

For more information, go to *Configuring a Demand-Based Scaling Policy*.

Understanding Health Checks

Auto scaling periodically performs *health checks* on the instances in your Auto Scaling group. By default, Auto Scaling group determines the health state of each instance by periodically checking the results of Eucalyptus instance status checks. When Auto Scaling determines that an instance is unhealthy, it terminates that instance and launches a new one.

If your Auto Scaling group is using elastic load balancing, Auto Scaling can determine the health status of the instances by checking the results of both Eucalyptus instance status checks and elastic load balancing instance health.

Auto scaling determines an instance is unhealthy if the calls to either Eucalyptus instance status checks or elastic load balancing instance health status checks return any state other than `OK` (or `InService`).

If there are multiple elastic load balancers associated with your Auto Scaling group, Auto Scaling will make health check calls to each load balancer. If any of the health check calls return any state other than `InService`, that instance will be marked as unhealthy. After Auto Scaling marks an instance as unhealthy, it will remain marked in that state, even if subsequent calls from other load balancers return an `InService` state for the same instance.

For more information, go to *Configuring Health Checks*.

Understanding Instance Termination Policies

Before Auto Scaling selects an instance to terminate, it first identifies the availability zone used by the group that contains the most instances. If all availability zones have the same number of instances, Auto Scaling selects a random availability zone, and then uses the termination policy to select the instance within that randomly selected availability zone for termination.

By default, Auto Scaling chooses the instance that was launched with the oldest launch configuration. If more than one instance was launched using the oldest launch configuration, Auto Scaling the instance that was created first using the oldest launch configuration.

You can override the default instance termination policy for your Auto Scaling group with one of the following options:

Option	Description
OldestInstance	The oldest instance in the Auto Scaling group should be terminated.
NewestInstance	The newest instance in the Auto Scaling group should be terminated.
OldestLaunchConfiguration	The first instance created using the oldest launch configration should be terminated.
Default	Use the default instance termination policy.

You can have more than one termination policy per Auto Scaling group. The termination policies are executed in the order they were created.

For more information, go to *Configuring an Instance Termination Policy*.

Auto Scaling Usage Examples

This section contains examples of many common usage scenarios for Auto Scaling.

- *Creating a Basic Auto Scaling Configuration*
- *Configuring a Demand-Based Scaling Policy*
- *Configuring Health Checks*
- *Configuring an Instance Termination Policy*

Creating a Basic Auto Scaling Configuration

Auto scaling requires two basic elements to function properly: a launch configuration and an Auto Scaling group. The following examples show how to set up the basic elements of an Auto Scaling configuraton.

Create a Launch Configuration

The launch configuration specifies the parameters that Auto Scaling will use when launching new instances for your Auto Scaling group. In this example, we will create a launch configuration with the minimum required parameters - in this case, we set up a launch configuration that specifies that all instances launched in the Auto Scaling group will be of instance type `m1.small` and use the `emi-00123456` image.

To create a launch configuration:

1. Enter the following command:
   ```
   euscale-create-launch-config MyLaunchConfig --image-id emi-00123456
   --instance-type m1.small
   ```
2. To verify the launch configuration, enter the following command:
   ```
   euscale-describe-launch-configs MyLaunchConfig
   ```

You should see output similar to the following:

```
LAUNCH-CONFIG MyLaunchConfig emi-00123456 m1.small
```

You've now created a launch configuration..

Create an Auto Scaling Group

The Auto Scaling group contains settings like the minimum, maximum, and desired number of Eucalyptus instances. At minimum, you must specify the name of the Auto Scaling group, the name of an existing launch configuration, the availability zone, and the minimum and maximum number of instances that should be running. In the following example, we will create an Auto Scaling group called MyScalingGroup in availability zone PARTI00, with a minimum size of 2 and a maximum size of 5.

To create an Auto Scaling group:

1. Enter the following command:

   ```
   euscale-create-auto-scaling-group MyScalingGroup --launch-configuration
   MyLaunchConfig --availability-zones PARTI00 --min-size 2 --max-size 5
   ```
2. Enter the following command to verify the Auto Scaling group you just created:

   ```
   euscale-describe-auto-scaling-groups MyScalingGroup
   ```

   This command will return output similar to the following:

   ```
   AUTO-SCALING-GROUP MyScalingGroup MyLaunchConfig PARTI00  2 5 2 Default
   INSTANCE i-1B853EC3 PARTI00 InService Healthy MyLaunchConfig
   INSTANCE i-ABC53ED7 PARTI00 InService Healthy MyLaunchConfig
   ```

Once you've created the Auto Scaling group, Auto Scaling will launch the appropriate number of instances.

Configuring a Demand-Based Scaling Policy

An Auto Scaling group needs a scaling policy to determine when to perform scaling activities. Auto scaling policies work with CloudWatch to identify metrics and set alarms, which are triggered when the metrics fall outside of a specified value range. To configure a scale-based policy, you need to create the policy, and then create CloudWatch alarms to associate with the policy.

In the following example, we will create a demand-based scaling policy.

Note: For more information on CloudWatch, go to *Using CloudWatch*.

Create Auto Scaling Policies

To create the scaling policies:

1. Create a scale-out policy using the following command:

   ```
   euscale-put-scaling-policy MyScaleoutPolicy --auto-scaling-group MyScalingGroup
    --adjustment=30 --type PercentChangeInCapacity
   ```

   This command will return a unique Amazon Resource Name (ARN) for the new policy.

   Note: You will need this ARN to create the Cloudwatch alarms in subsequent steps.

Note: The following example has been split into two lines for legibility.

```
arn:aws:autoscaling::706221218191:scalingPolicy:5d02981b-f440-4c8f-98f2-8a620dc2b787:

    autoScalingGroupName/MyScalingGroup:policyName/MyScaleoutPolicy
```

2. Create a scale-in policy using the following command:

```
euscale-put-scaling-policy MyScaleinPolicy -g MyScalingGroup --adjustment=-2
  --type ChangeInCapacity
```

This command will return output containing a unique Amazon Resource Name (ARN) for the new policy, similiar to the following:

Note: You will need this ARN to create the Cloudwatch alarms in subsequent steps.

Note: The following example has been split into two lines for legibility.

```
arn:aws:autoscaling::706221218191:scalingPolicy:d28c6ffc-e9f1-4a48-a79c-8b431794c367:

    autoScalingGroupName/MyScalingGroup:policyName/MyScaleinPolicy
```

Create CloudWatch Alarms

To create CloudWatch alarms:

1. Create a Cloudwatch alarm for scaling out when the average CPU usage exceeds 80 percent:

Note: The following example has been split into multiple lines for legibility.

```
euwatch-put-metric-alarm AddCapacity  --metric-name CPUUtilization --namespace
 "AWS/EC2"
--statistic Average --period 120 --threshold 80 --comparison-operator
GreaterThanOrEqualToThreshold --dimensions
"AutoScalingGroupName=MyScalingGroup"
--evaluation-periods 2
--alarm-actions
arn:aws:autoscaling::706221218191:scalingPolicy:5d02981b-f440-4c8f-98f2-8a620dc2b787:

   autoScalingGroupName/MyScalingGroup:policyName/MyScaleoutPolicy
```

2. Create a Cloudwatch alarm for scaling in when the average CPU usage falls below 40 percent:

Note: The following example has been split into multiple lines for legibility.

```
euwatch-put-metric-alarm RemoveCapacity --metric-name CPUUtilization
--namespace "AWS/EC2"
 --statistic Average  --period 120  --threshold 40  --comparison-operator
LessThanOrEqualToThreshold  --dimensions "AutoScalingGroupName=MyScalingGroup"

 --evaluation-periods 2
 --alarm-actions
arn:aws:autoscaling::706221218191:scalingPolicy:d28c6ffc-e9f1-4a48-a79c-8b431794c367:

  autoScalingGroupName/MyScalingGroup:policyName/MyScaleinPolicy
```

Verify Your Alarms and Policies

Once you've created your Auto Scaling policies and CloudWatch alarms, you should verify them.

To verify your alarms and policies:

1. Use the CloudWatch command `euwatch-describe-alarms` to see a list of the alarms you've created:

```
euwatch-describe-alarms
```

   This will return output similar to the following:

```
AddCapacity INSUFFICIENT_DATA
arn:aws:autoscaling::706221218191:scalingPolicy:5d02981b-f440-4c8f-98f2-8a620dc2b787:

   autoScalingGroupName/MyScalingGroup:policyName/MyScaleoutPolicy
   AWS/EC2 CPUUtilization 120 Average 2 GreaterThanOrEqualToThreshold 80.0
RemoveCapacity INSUFFICIENT_DATA
arn:aws:autoscaling::706221218191:scalingPolicy:d28c6ffc-e9f1-4a48-a79c-8b431794c367:
autoScalingGroupName/MyScalingGroup:policyName/MyScaleinPolicy
   AWS/EC2 CPUUtilization 120 Average 2 LessThanOrEqualToThreshold 40.0
```

2. Use the euscale-describe-policies command to see the scaling policies you've created:

```
euscale-describe-policies --auto-scaling-group MyScalingGroup
```

   This will return output similar to the following (note that this output has been split into multiple lines for legibility):

```
SCALING-POLICY MyScalingGroup MyScaleinPolicy -2 ChangeInCapacity
arn:aws:autoscaling::706221218191:
  scalingPolicy:d28c6ffc-e9f1-4a48-a79c-8b431794c367:
  autoScalingGroupName/MyScalingGroup:policyName/MyScaleinPolicy
SCALING-POLICY MyScalingGroup MyScaleoutPolicy 30 PercentChangeInCapacity
arn:aws:autoscaling::706221218191:
  scalingPolicy:5d02981b-f440-4c8f-98f2-8a620dc2b787:
  autoScalingGroupName/MyScalingGroup:policyName/MyScaleoutPolicy
```

Configuring Health Checks

By default, Auto Scaling group determines the health state of each instance by periodically checking the results of instance status checks. You can specify using the ELB health check method in addition to using the instance health check method.

To use load balancing health checks for an Auto Scaling group:

Use the following command to specify ELB health checks for an Auto Scaling group:

```
euscale-update-auto-scaling-group MyScalingGroup –-health-check-type ELB
–-grace-period 300
```

Configuring an Instance Termination Policy

You can control how Auto Scaling determines which instances to terminate. You can specify a termination policy when you create an Auto Scaling group, and you can change the termination policy at any time using the `euscale-update-auto-scaling-group` command.

You can override the default instance termination policy for your Auto Scaling group with one of the following options:

Option	Description
OldestInstance	The oldest instance in the Auto Scaling group should be terminated.
NewestInstance	The newest instance in the Auto Scaling group should be terminated.
OldestLaunchConfiguration	The first instance created using the oldest launch configration should be terminated.
Default	Use the default instance termination policy.

To configure an instance termination policy:

1. Specify the --termination-policies parameter when creating or updating the Auto Scaling group. For example:

```
euscale-update-auto-scaling-group MyScalingGroup --termination-policies
"NewestInstance"
```

2. Verify that your Auto Scaling group has updated the termination policy by running the following command:
   `euscale-describe-auto-scaling-groups MyScalingGroup`

   This command should return output similar to the following:

```
AUTO-SCALING-GROUP MyScalingGroup MyLaunchConfig PARTI00  2 5 2 NewestInstance
INSTANCE i-1B853EC3 PARTI00 InService Healthy MyLaunchConfig
INSTANCE i-ABC53ED7 PARTI00 InService Healthy MyLaunchConfig
```

Using Elastic Load Balancing

Elastic Load Balancing automatically distributes incoming traffic across your Eucalyptus cloud. It enables you to achieve even greater fault tolerance by automatically detecting instances that are overloaded, and rerouting traffic to other instances as needed.

Elastic Load Balancing works in conjunction with Eucalyptus's Auto Scaling and Cloud Watch services, to make your cloud more robust and efficient.

The following section describes how load balancing works, provides an overview of load balancing concepts, and includes a series of usage scenarios that will show you how to perform common load balancing tasks.

Elastic Load Balancing Overview

Elastic Load Balancing is designed to provide an easy-to-use fault tolerant cloud platform. Using Elastic Load Balancing, you can automatically balance incoming traffic, ensuring that requests are sent to an instance that has the capacity to serve them. It allows you to add and remove instances as needed, without interrupting the operation of your cloud. If an instance fails or is removed, the load balancer stops routing traffic to that instance. If that instance is restored, or an instance is added, the load balancer automatically resumes or starts routing traffic to that instance.

If your cloud is serving traffic over HTTP, the Elastic Load Balancing service can provide session stickiness, allowing you to bind specific virtual instances to your load balancers.

Elastic Load Balancing can balance requests within a cluster, or across multiple clusters. If there are no more healthy instances in a cluster, the load balancer will route traffic to other clusters, until healthy instances are restored to that cluster.

Elastic Load Balancing Concepts

This section describes the terminology and concepts you need for understanding and using the Elastic Load Balancing service.

DNS

When a new load balancer is created, Eucalyptus will assign a unique DNS A record to the load balancer. After the load balancer has been launched and configured, the IP address of its interface will be added to the DNS name. If more than one cluster is specified, there can be more than one IP addresses mapped to the DNS name.

The load balancers in one cluster will distribute traffic to the instances only in the same cluster. Application users can query the application service using the DNS name and the Eucalyptus DNS service will respond to the query with the list of IP addresses on a round-robin basis.

Eucalyptus generates a DNS name automatically (e.g., lb001.euca-cloud.example.com). Typically there will also be a CNAME record that maps from the meaningful domain name (e.g., service.example.com) to the automatically-generated DNS name. The CNAME records can be serviced anywhere on Internet.

Health Check

In order to route traffic to healthy instances, and prevent traffic from being routed to unhealthy instances, the Elastic Load Balancing service routinely checks the health of your service instances. The health check uses metrics such as latency, RequestCount and HTTP response code counts to ensure that service instances are responding appropriately to user traffic.

Load Balancer

Load balancers are the core of the Elastic Load Balancing service. Load balancers are special instances created from a Eucalyptus-provided VM image, and are managed by Eucalyptus. Instances off the image forward packets to your service instances using load-balancing software to ensure no instances are overloaded.

There may be a brief delay between the time that the first service instance is registered to a load balancer and the time the load balancer becomes operational. This is due to the virtual machine instantiation, and will not adversely affect the operation of your load balancer.

Each balancer VM will service only one load balancer. Launching unnecessary load balancers may be detrimental to your cloud's performance.

Service Instance

A service instance is an instance created from an image in your cloud. These are the instances that serve your application and users. The Elastic Load Balancing service monitors the health of these instances and routes traffic only to healthy instances.

Eucalyptus Load Balancing Usage Examples

This section contains examples of common usage scenarios for Eucalyptus Load Balancing.

Creating a Basic Elastic Load Balancing Configuration

Elastic load balancing requires two basic elements to function properly: a load balancer and instances registered with that load balancer. The following examples show how to set up the basic elements of an elastic load balancer configuraton.

Create a Load Balancer

The load balancer manages incoming traffic, and monitors the health of your instances. The load balancer ensures that traffic is only sent to healthy instances.

To create a load balancer:

1. Enter the following command, specifying availability zones:

```
eulb-create-lb -z PARTI00 -l "lb-port=80, protocol=HTTP, instance-port=80,
instance-protocol=HTTP" MyLoadBalancer
```

2. To verify the elastic load balancer has been created, enter the following command:
   `eulb-describe-lbs MyLoadBalancer`

   You should see output similar to the following:

```
LOAD_BALANCER MyLoadBalancer MyLoadBalancer-587773761872.lb.localhost
2013-01-01T01:23:45.678Z
```

3. You must now create listeners for the load balancer as follows:

```
eulb-create-lb-listeners --listener "lb-port=PORT, protocol=PROTOCOL,
                                     instance-port=PORT,
instance-protocol=PROTOCOL,
                                     cert-id=ARN"
                        ELB
```

You've now created an elastic load balancer.

Register instances with the Load Balancer

The load balancer monitors the health of registered instances, and balances incoming traffic across the healthy instances.

To register an instance with the load balancer:

1. Enter the following command:

```
eulb-register-instances-with-lb
  --instances INSTANCE1,INSTANCE2,...     \
 (--region USER@REGION | --url URL)       \
  --access-key-id KEY_ID --secret-key KEY \
    MyLoadBalancer
```

2. Enter the following command to verify that the instances are registered with the load balancer:

   `eulb-describe-instance-health MyLoadBalancer`

   This command will return output similar to the following:

```
INSTANCE i-6FAD3F7B InService
INSTANCE i-70FE4541 InService
```

Once you've created the load balancer and registered your instances with it, the load balancer will automatically route traffic from its endpoint URL to healthy instances.

Configuring the Health Check

To determine which instances are healthy, the load balancer periodically polls the registered instances. You can use the `eulb-configure-healthcheck` command as described below to configure how the instances should be polled, how long to wait for a response, and how many consecutive successes or failures are required to mark an instance as healthy or unhealthy.

Enter the following command:

```
eulb-configure-healthcheck
  --healthy-threshold   COUNT   \
  --unhealthy-threshold COUNT   \
  --interval SECONDS            \
  --timeout  SECONDS            \
  --target PROTOCOL:PORT[/PATH] \
    MyLoadBalancer
```

Use `--healthy-threshold` and `--unhealthy-threshold` to specify the number of consecutive health checks required to mark an instance as Healthy or Unhealthy respectively. Use `--target` to specify the connection target on your instances for these health checks. Use `--interval` and `--timeout` to specify the approximate frequency and maximum duration of these health checks.

Modifying an Elastic Load Balancing Configuration

Elastic load balancing requires two basic elements to function properly: a load balancer and instances registered with that load balancer. The following examples show how to modify the basic elements of an elastic load balancer configuraton.

De-register instances from the Load Balancer

The load balancer monitors the health of registered instances, and balances incoming traffic across the healthy instances.

To deregister an instance from the load balancer:

1. Enter the following command:

```
eulb-deregister-instances-from-lb --instances INSTANCE1,INSTANCE2,...
MyLoadBalancer
```

2. Enter the following command to verify that the instances are deregistered from the load balancer:

```
eulb-describe-instance-health MyLoadBalancer
```

This command will return output similar to the following:

```
INSTANCE i-6FAD3F7B InService
```

Delete Load Balancer Listeners

To delete a load balancer listener:

Enter the following command:

```
eulb-delete-lb-listeners --lb-ports PORT1,PORT2,... MyLoadBalancer
```

Delete Load Balancer

To delete a load balancer:

Enter the following command:

```
eulb-delete-lb MyLoadBalancer
```

You've now deleted the elastic load balancer.

Using CloudWatch

CloudWatch is a Eucalyptus service that enables you to monitor, manage, and publish various metrics, as well as configure alarm actions based on data from metrics. You can use the default metrics that come with Eucalyptus, or you can use your own metrics.

CloudWatch collects raw data from your cloud's resources and generates the information into readable, near real-time metrics. These metrics are recorded for a period of two weeks. This allows you to access historical information and provides you with information about how your resource is performing.

To find out more about what CloudWatch is, see *CloudWatch Overview*.

To find out how to use CloudWatch, see *CloudWatch Tasks*.

CloudWatch Overview

This section describes the concepts and details you need to understand the CloudWatch service. This section also includes procedures to complete the most common tasks for CloudWatch.

CloudWatch is Eucalyptus services that collects, aggregates, and dispenses data from your cloud's resources. This data allows you to make operational and business decisions based on actual performance metrics. You can use CloudWatch to collect metrics about your cloud resources, such as the performance of your instances. You can also publish your own metrics directly to CloudWatch.

CloudWatch monitors the following cloud resources:

- instances
- Elastic Block Store (EBS) volumes
- Auto Scaling instances
- Eucalyptus Load Balancers (ELB)

Alarms

CloudWatch alarms help you make decisions more easily by automatically making changes to the resources you are monitoring, based on rules that you define. For example, you can create alarms that initiate Auto Scaling actions on your behalf. For more information about alarm tasks, see *Configuring Alarms*.

Common Use Cases

A common use for CloudWatch is to keep your applications and services healthy and running efficiently. For example, CloudWatch can determine that your application runs best when network traffic remains below a certain threshold level on your instances. You can then create an automated procedure to ensure that you always have the right number of instances to match the amount of traffic you have.

Another use for CloudWatch is to diagnose problems by looking at system performance before and after a problem occurs. CloudWatch helps you identify the cause and verify your fix by tracking performance in real time.

CloudWatch Concepts

This section describes the terminology and concepts you need in order to understand and use CloudWatch.

Metric

A metric is a time-ordered set of data points. You can get metric data from Eucalyptus cloud resources (like instances or volumes), or you can publish your own set of custom metric data points to CloudWatch. You then retrieve statistics about those data points as an ordered set of time-series data.

Data points represent values of a variable over time. For example you can get metrics for the CPU usage of a particular instance, or for the latency of an elastic load balancer (ELB).

Each metric is uniquely defined by a name, a namespace, and zero or more dimensions. Each data point has a time stamp, and (optionally) a unit of measure. When you request statistics, the returned data stream is identified by namespace, metric name, dimension, and (optionally) the unit. For more information about Eucalyptus-supported metrics, see *Namespaces, Metrics, and Dimensions*.

CloudWatch stores your metric data for two weeks. You can publish metric data from multiple sources, such as incoming network traffic from dozens of different instances, or requested page views from several different web applications. You can request statistics on metric data points that occur within a specified time window.

Namespace

A namespace is a conceptual container for a collection of metrics. Eucalyptus treats metrics in different namespaces as unique. This means that metrics from different services cannot mistakenly be aggregated into the same statistical set.

Namespace names are strings you define when you create a metric. The names must be valid XML characters, typically containing the alphanumeric characters "0-9A-Za-z" plus "."(period), "-" (hyphen), "_" (underscore), "/" (slash), "#" (hash), and ":" (colon). AWS namespaces all follow the convention AWS/<service>, such as AWS/EC2 and AWS/ELB. For more information, see *Namespaces*.

Note: A namespace name must be less than 256 characters. There is no default namespace. You must specify a namespace for each data element you put into CloudWatch.

Dimension

A dimension is a name-value pair that uniquely identifies a metric. A dimension helps you design a conceptual structure for your statistics plan. Because dimensions are part of the unique identifier for a metric, metric name, namespace, and dimension key-value pairs define unique metrics.

You specify dimensions when you create a metric with the `euwatch-put-data` command. Eucalyptus services that report data to CloudWatch also attach dimensions to each metric. You can use dimensions to filter result sets that CloudWatch queries return. For example, you can get statistics for a specific instance by calling `euwatch-get-stats` with the InstanceID dimension set to a specific instance ID.

For Eucalyptus metrics, CloudWatch can aggregate data across select dimensions. For example, if you request a metric in the AWS/EC2 namespace and do not specify any dimensions, CloudWatch aggregates all data for the specified metric to create the statistic that you requested. However, CloudWatch does not aggregate across dimensions for custom metrics

Note: You can assign up to ten dimensions to a metric.

Time Stamp

Each metric data point must be marked with a time stamp. The time stamp can be up to two weeks in the past and up to two hours into the future. If you do not provide a time stamp, CloudWatch creates a time stamp for you based on the time the data element was received.

The time stamp you use in the request must be a dateTime object, with the complete date plus hours, minutes, and seconds. For example: 2007-01-31T23:59:59Z. For more information, go to *http://www.w3.org/TR/xmlschema-2/#dateTime*. Although it is not required, we recommend you provide the time stamp in the Coordinated Universal Time (UTC or Greenwich Mean Time) time zone. When you retrieve your statistics from CloudWatch, all times reflect the UTC time zone.

Unit

A unit represents a statistic's measurement in time or amount. For example, the units for the instance NetworkIn metric is Bytes because NetworkIn tracks the number of bytes that an instance receives on all network interfaces.

You can also specify a unit when you create a custom metric. Units help provide conceptual meaning to your data. Metric data points you create that specify a unit of measure, such as Percent, will be aggregated separately. The following list provides some of the more common units that CloudWatch supports:

- Seconds
- Bytes
- Bits
- Percent
- Count
- Bytes/Second (bytes per second)
- Bits/Second (bits per second)
- Count/Second (counts per second)
- None (default when no unit is specified)

Though CloudWatch attaches no significance to a unit, other applications can derive semantic information based on the unit you choose. When you publish data without specifying a unit, CloudWatch associates it with the None unit. When you get statistics without specifying a unit, CloudWatch aggregates all data points of the same unit together. If you have two otherwise identical metrics with different units, two separate data streams will be returned, one for each unit.

Statistic

A statistic is computed aggregation of metric data over a specified period of time. CloudWatch provides statistics based on the metric data points you or Eucalyptus provide. Aggregations are made using the namespace, metric name, dimensions, and the data point unit of measure, within the time period you specify. The following table describes the available statistics.

Statistic	Description
`Minimum`	The lowest value observed during the specified period. You can use this value to determine low volumes of activity for your application.
`Maximum`	The highest value observed during the specified period. You can use this value to determine high volumes of activity for your application.
`Sum`	All values submitted for the matching metric added together. You can use this statistic for determining the total volume of a metric.
`Average`	The value of Sum / SampleCount during the specified period. By comparing this statistic with the Minimum and Maximum, you can determine the full scope of a metric and how close the average use is to the Minimum and Maximum. This comparison helps you to know when to increase or decrease your resources as needed.
`SampleCount`	The count (number) of data points used for the statistical calculation.

Period

A period is the length of time, in seconds, associated with a specific CloudWatch statistic. Each statistic represents an aggregation of the metrics data collected for a specified period of time. You can adjust how the data is aggregated by varying the length of the period. A period can be as short as one minute (60 seconds) or as long as two weeks (1,209,600 seconds).

The values you select for the StartTime and EndTime options determine how many periods CloudWatch returns. For example, if you set values for the Period, EndTime, and StartTime options for 60 seconds, CloudWatch returns an aggregated set of statistics for each minute of the previous hour. If you want statistics aggregated into ten-minute blocks, set Period to 600. For statistics aggregated over the entire hour, use a Period value of 3600.

Periods are also an important part of the CloudWatch alarms feature. When you create an alarm to monitor a specific metric, you are asking CloudWatch to compare that metric to the threshold value that you supplied. You have control over how CloudWatch makes that comparison. You can specify the period over which the comparison is made, as well as how many consecutive periods the threshold must be breached before you are notified.

Aggregation

CloudWatch aggregates statistics according a length of time that you set. You can publish as many data points as you want with the same or similar time stamps. CloudWatch aggregates these data points by period length. You can publish data points for a metric that share not only the same time stamp, but also the same namespace and dimensions.

Subsequent calls to `euwatch-get-stats` return aggregated statistics about those data points. You can even do this in one `euwatch-put-data` request. CloudWatch accepts multiple data points in the same `euwatch-put-data` call with the same time stamp. You can also publish multiple data points for the same or different metrics, with any time stamp. The size of a `euwatch-put-data` request, however, is limited to 8KB for HTTP GET requests and 40KB for HTTP POST requests. You can include a maximum of 20 data points in one PutMetricData request.

For large data sets that would make the use of `euwatch-put-data` impractical, CloudWatch allows you to insert a pre-aggregated data set called a StatisticSet. With StatisticSets you give CloudWatch the Min, Max, Sum, and SampleCount of a number of data points. A common use case for StatisticSets is when you are collecting data many times in a minute. For example, if you have a metric for the request latency of a server, it doesn’t make sense to do a `euwatch-put-data` request with every request. We suggest you collect the latency of all hits to that server, aggregate them together once a minute and send that StatisticSet to CloudWatch.

CloudWatch doesn't differentiate the source of a metric. If you publish a metric with the same namespace and dimensions from different sources, CloudWatch treats this as a single metric. This can be useful for service metrics in a distributed, scaled system. For example, all the hosts in a web server application could publish identical metrics representing the latency of requests they are processing. CloudWatch treats these as a single metric, allowing you to get the statistics for minimum, maximum, average, and sum of all requests across your application.

Alarm

An alarm watches a single metric over a time period you set, and performs one or more actions based on the value of the metric relative to a given threshold over a number of time periods. CloudWatch alarms will not invoke actions just because they are in a particular state. The state must have changed and been maintained for a specified number of periods. After an alarm invokes an action due to a change in state, the alarm continues to invoke the action for every period that the alarm remains in the new state.

An alarm has three possible states:

- `OK`: The metric is within the defined threshold.
- `ALARM`: The metric is outside of the defined threshold.
- `INSUFFICIENT_DATA`: The alarm has just started, the metric is not available, or not enough data is available for the metric to determine the alarm state.

The following lists some common features of alarms:

- You can create up to 5000 alarms per Eucalyptus account. To create or update an alarm, use the `euwatch-put-metric-alarm` command.
- You can list any or all of the currently configured alarms, and list any alarms in a particular state using the `euwatch-describe-alarms` command.
- You can disable and enable alarms by using the `euwatch-disable-alarm-actions` and `euwatch-enable-alarm-actions` commands
- You can test an alarm by setting it to any state using the `euwatch-set-alarm-state` command. This temporary state change lasts only until the next alarm comparison occurs.
- Finally, you can view an alarm's history using the `euwatch-describe-alarm-history` command. CloudWatch preserves alarm history for two weeks. Each state transition is marked with a unique time stamp. In rare cases, your

history might show more than one notification for a state change. The time stamp enables you to confirm unique state changes.

Namespaces, Metrics, and Dimensions

This section discusses the namespaces, metrics, and dimensions that CloudWatch supports for Eucalyptus services.

Namespaces

All Eucalyptus services that provide CloudWatch data use a namespace string, beginning with "AWS/". This section describes the service namespaces.

The following table lists the namespaces for services that push metric data points to CloudWatch.

Service	Namespace
Elastic Block Store	AWS/EBS
Elastic Compute Cloud	AWS/EC2
Auto Scaling	AWS/Autoscaling
Elastic Load Balancing	AWS/ELB

Instance Metrics and Dimensions

This section describes the instance metrics and dimensions available to CloudWatch.

Available Metrics for Instances

Metric	Description	Unit
`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.	Percent
`DiskReadOps`	Completed read operations from all ephemeral disks available to the instance (if your instance uses EBS, see EBS Metrics.) 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.	Count
`DiskWriteOps`	Completed write operations to all ephemeral disks available to the instance (if your instance uses Amazon EBS, see Amazon EBS Metrics.) 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.	Count

Metric	Description	Unit
`DiskReadBytes`	Bytes read from all ephemeral disks available to the instance (if your instance uses Amazon EBS, see Amazon EBS Metrics.) 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.	Bytes
`DiskWriteBytes`	Bytes written to all ephemeral disks available to the instance (if your instance uses Amazon EBS, see Amazon EBS Metrics.) 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.	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.	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.	Bytes

Available Dimensions for Instances

You can filter the instance data using any of the dimensions in the following table.

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 instance metrics when the instances are in such an Auto Scaling group.
`ImageId`	This dimension filters the data you request for all instances running this Eucalyptus Machine Image (EMI).
`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.

Dimension	Description
`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.

EBS Metrics and Dimensions

This section describes the Elastic Block Store (EBS) metrics and dimensions available to CloudWatch.

Available Metrics for EBS

Metric	Description	Unit
`VolumeReadBytes`	The total number of bytes transferred in the period.	Bytes
`VolumeWriteBytes`	The total number of bytes transferred in the period.	Bytes
`VolumeReadOps`	The total number of operations in the period.	Count
`VolumeWriteOps`	The total number of operations in the period.	Count
`VolumeTotalReadTime`	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.	Seconds
`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.	Seconds
`VolumeIdleTime`	The total number of seconds in the period when no read or write operations were submitted.	Seconds
`VolumeQueueLength`	The number of read and write operation requests waiting to be completed in the period.	Count

Available Dimensions for EBS

The only dimension that EBS sends to CloudWatch is the Volume ID. This means that all available statistics are filtered by Volume ID.

Auto Scaling Metrics and Dimensions

This section discusses the Auto Scaling metrics and dimensions available to CloudWatch.

Available Metrics for Auto Scaling

Metric	Description	Unit
`GroupMinSize`	The minimum size of the Auto Scaling group.	Count
`GroupMaxSize`	The maximum size of the Auto Scaling group.	Count
`GroupDesiredCapacity`	The number of instances that the Auto Scaling group attempts to maintain.	Count
`GroupInServiceInstances`	The number of instances that are running as part of the Auto Scaling group. This metric does not include instances that are pending or terminating.	Count
`GroupPendingInstances`	The number of instances that are pending. A pending instance is not yet in service. This metric does not include instances that are in service or terminating.	Count
`GroupTerminatingInstances`	The number of instances that are in the process of terminating. This metric does not include instances that are in service or pending.	Count
`GroupTotalInstances`	The total number of instances in the Auto Scaling group. This metric identifies the number of instances that are in service, pending, and terminating.	Count

Available Dimensions for Auto Scaling

The only dimension that Auto Scaling sends to CloudWatch is the name of the Auto Scaling group. This means that all available statistics are filtered by Auto Scaling group name.

ELB Metrics and Dimensions

This section discusses the Elastic Load Balancing (ELB) metrics and dimensions available to CloudWatch.

Available Metrics for ELB

Metric	Description	Unit
`Latency`	Time elapsed after the request leaves the load balancer until it receives the corresponding response. Valid Statistics: `Minimum` | `Maximum` | `Average` | `Count`	Seconds
`RequestCount`	The number of requests handled by the load balancer.	Count
`HealthyHostCount`	The number of healthy instances registered with the load balancer in a specified availability zone. Healthy instances are those that have not failed more health checks than the value of the unhealthy threshold. Constraints: You must provide both `LoadBalancerName` and `AvailabilityZone` dimensions for this metric. Valid Statistics: `Minimum` | `Maximum` | `Average`	Count
`UnHealthyHostCount`	The number of unhealthy instances registered with the load balancer. These are instances that have failed more health checks than the value of the unhealthy threshold. Constraints: You must provide both `LoadBalancerName` and `AvailabilityZone` dimensions for this metric. Valid Statistics: `Minimum` | `Maximum` | `Average`	Count
`HTTPCode_ELB_4XX`	Count of HTTP response codes generated by ELB that are in the 4xx (client error) series. Valid Statistics: `Sum`	Count
`HTTPCode_ELB_5XX`	Count of HTTP response codes generated by ELB that are in the 5xx (server error) series. ELB can generate 5xx errors if no back-end instances are registered, no healthy back-end instances, or the request rate exceeds ELB's current available capacity. This response count does not include any responses that were generated by back-end instances. Valid Statistics: `Sum`	Count

Metric	Description	Unit
`HTTPCode_Backend_2XX`	Count of HTTP response codes generated by back-end instances that are in the 2xx (success) series. Valid Statistics: `Sum`	Count
`HTTPCode_Backend_3XX`	Count of HTTP response codes generated by back-end instances that are in the 3xx (user action required) series. Valid Statistics: `Sum`	Count
`HTTPCode_Backend_4XX`	Count of HTTP response codes generated by back-end instances that are in the 4xx (client error) series. This response count does not include any responses that were generated by ELB. Valid Statistics: `Sum`	Count
`HTTPCode_Backend_5XX`	Count of HTTP response codes generated by back-end instances that are in the 5xx (server error) series. This response count does not include any responses that were generated by ELB. Valid Statistics: `Sum`	Count

Available Dimensions for ELB

You can use the currently available dimensions for ELB to refine the metrics returned by a query. For example, you could use `HealthyHostCount` and dimensions `LoadBalancerName` and `AvailabilityZone` to get the average number of healthy instances behind the specified load balancer within the specified Availability Zone for a given period of time.

You can aggregate ELB data along any of the following dimensions shown in the following table.

Metric	Description
`LoadBalancerName`	Limits the metric data to instances that are connected to the specified load balancer.
`AvailabilityZone`	Limits the metric data to load balancers in the specified availability zone.

CloudWatch Tasks

This section details the tasks you can perform using CloudWatch.

This section expands on the basic concepts presented in the preceding section (see *CloudWatch Overview* and includes procedures for using CloudWatch. This section also shows you how to view metrics that Eucalyptus services provide to CloudWatch and how to publish custom metrics with CloudWatch.

Configuring Monitoring

This section describes how to enable and disable monitoring for your cloud resources.

Enable Monitoring

This section describes steps for enabling monitoring on your cloud resources.

To enable monitoring on your resources, following the steps for your resource.

Enable monitoring for an instance

- To enable monitoring for a running instance, enter the following command:

```
euca-monitor-instances [instance_id]
```

- To enable monitoring when you launch an instance, enter the following command:

```
euca-run-instances [image_id] -k gsg-keypair --monitor
```

Enable monitoring for a scaling group

- To enable monitoring for an existing Auto Scaling group:
  a) Create a launch configuration with `--monitoring-enabled` option.
  b) Make a `euscale-update-auto-scaling-group` request to update your Auto Scaling group with the launch configuration you created in the previous step. Auto Scaling will enable monitoring for new instances that it creates.
  c) Choose one of the following actions to deal with all existing instances in the Auto Scaling group:

To . . .	Do this . . .
Preserve existing instances	Make a `euca-monitor-instances` request for all existing instances to enable monitoring.
Terminate existing instances	Make a `euscale-terminate-instance-in-auto-scaling-group` request for all existing instances. Auto Scaling will use the updated launch configuration to create replacement instances with monitoring enabled.

- To enable monitoring when you create a new Auto Scaling group:
  a) Create a launch configuration with `--monitoring-enabled` option.

Enable monitoring for a load balancer

Elastic Load Balancing (ELB) sends metrics and dimensions for all load balancers to CloudWatch. By default, you do not need to specifically enable monitoring.

Important: ELB only sends CloudWatch metrics when requests are sent through the load balancer. If there are no requests or data for a given metric, ELB does not report to CloudWatch. If there are requests sent through the load balancer, ELB measures and sends metrics for that load balancer in 60-second intervals.

Disable Monitoring

This section describes steps for disabling monitoring on your cloud resources.

To disable monitoring on your resources, following the steps for your resource.

Disable monitoring for an instance

- To disable monitoring for a running instance, enter the following command:

```
euca-unmonitor-instances [instance_id]
```

Disable monitoring for a scaling group

- To enable monitoring for an existing Auto Scaling group:
  a) Create a launch configuration with `--monitoring-disabled` option.
  b) Make a `euscale-update-auto-scaling-group` request to update your Auto Scaling group with the launch configuration you created in the previous step. Auto Scaling will disable monitoring for new instances that it creates.
  c) Choose one of the following actions to deal with all existing instances in the Auto Scaling group:

To . . .	Do this . . .
Preserve existing instances	Make a `euca-unmonitor-instances` request for all existing instances to disable monitoring.
Terminate existing instances	Make a `euscale-terminate-instance-in-auto-scaling-group` request for all existing instances. Auto Scaling will use the updated launch configuration to create replacement instances with monitoring disabled.

- To enable monitoring when you create a new Auto Scaling group:
  a) Create a launch configuration with `--monitoring-disabled` option.

Disable monitoring for a load balancer

There is no way to disable monitoring for a load balancer.

Viewing and Publishing Metrics

This section describes how to view Eucalyptus metrics as well as how to publish your own metrics.

List Available Metrics

To list available metrics:

Enter the following command.

```
euwatch-list-metrics
```

Eucalyptus returns a listing of all metrics, as shown in the following partial example output:

```
Metric Name         Namespace  Dimensions
CPUUtilization      AWS/EC2    {InstanceId=i-5431413d}
CPUUtilization      AWS/EC2    {InstanceId=i-d43242bd}
CPUUtilization      AWS/EC2    {InstanceId=i-1d3d4d74}
CPUUtilization      AWS/EC2    {InstanceId=i-78314111}
CPUUtilization      AWS/EC2    {InstanceId=i-d3c8baba}
CPUUtilization      AWS/EC2    {InstanceId=i-0d334364}
CPUUtilization      AWS/EC2    {InstanceId=i-6732420e}
CPUUtilization      AWS/EC2    {InstanceId=i-d93141b0}
CPUUtilization      AWS/EC2    {InstanceId=i-e03d4d89}
CPUUtilization      AWS/EC2    {InstanceId=i-c93d4da0}
CPUUtilization      AWS/EC2    {InstanceId=i-e0304089}
CPUUtilization      AWS/EC2    {InstanceId=i-e1304088}
CPUUtilization      AWS/EC2    {InstanceId=i-69334300}
```

Get Statistics for a Metric

To get statistics for a metric:

Enter the following command.

```
euwatch-get-stats [metric_name] --start-time [start_time] --end-time [end_time]

               --period [time_in_seconds] --namespace "AWS/EC2" --statistics
 [statistic] --dimensions
               [dimension] --headers
```

The following example returns the average CPU utilization for the i-c08804a9 instance at one hour resolution.

```
euwatch-get-stats CPUUtilization --start-time
2013-02-14T23:00:00.000Z
--end-time 2013-03-14T23:00:00.000Z --period 3600 --statistics
"Average"
--namespace "AWS/EC2" --dimensions "InstanceId=i-c08804a9"
```

The following example returns CPU utilization for all of your cloud's instances.

```
euwatch-get-stats CPUUtilization --start-time
2013-02-14T23:00:00.000Z --end-time
2013-03-14T23:00:00.000Z --period 3600 --statistics
"Average,Minimum,Maximum" --namespace "AWS/EC2"
```

Publish Custom Metrics

CloudWatch allows you to publish your own metrics, such as application performance, system health, or customer usage.

Publish a single data point

To publish a single data point for a new or existing metric, call mon-put-data with one value and time stamp. For example, the following actions each publish one data point:

```
euwatch-put-data --metric-name PageViewCount --namespace "TestService" --value
 2 --timestamp 2011-03-14T12:00:00.000Z
euwatch-put-data --metric-name PageViewCount --namespace "TestService" --value
 4 --timestamp 2011-03-14T12:00:01.000Z
euwatch-put-data --metric-name PageViewCount --namespace "TestService" --value
 5 --timestamp 2011-03-14T12:00:02.000Z
```

You can publish data points with time stamps as granular as one-thousandth of a second. However, CloudWatch aggregates the data to a minimum granularity of 60 seconds. For example, the `PageViewCount` metric from the previous examples contains three data points with time stamps just seconds apart. CloudWatch aggregates the three data points because they all have time stamps within a 60-second period.

CloudWatch uses 60-second boundaries when aggregating data points. For example, CloudWatch aggregates the data points from the previous example because all three data points fall within the 60-second period that begins at 2011-03-14T12:00:00.000Z and ends at 2011-03-14T12:00:59.999Z.

Publish statistic sets

You can also aggregate your data before you publish to CloudWatch. When you have multiple data points per minute, aggregating data minimizes the number of calls to `euwatch-put-data`. For example, instead of calling `euwatch-put-data` multiple times for three data points that are within three seconds of each other, you can aggregate the data into a statistic set that you publish with one call:

```
euwatch-put-data --metric-name PageViewCount --namespace "TestService" -s
"Sum=11,Minimum=2,Maximum=5,SampleCount=3" --timestamp 2011-03-14T12:00:00.000
```

Publish the value zero

When your data is more sporadic and you have periods that have no associated data, you can choose to publish the value zero (0) for that period or no value at all. You might want to publish zero instead of no value if you use periodic calls to PutMetricData to monitor the health of your application. For example, you can set an Amazon CloudWatch alarm to notify you if your application fails to publish metrics every five minutes. You want such an application to publish zeros for periods with no associated data.

You might also publish zeros if you want to track the total number of data points or if you want statistics such as minimum and average to include data points with the value 0.

Configuring Alarms

This section describes how to create, test, and delete and alarm.

Create an Alarm

You can create a CloudWatch alarm using a resource's metric, and then add an action using the action's dedicated Amazon Resource Name (ARN). You can add the action to any alarm state.

Important: Eucalyptus currently only supports actions for executing Auto Scaling policies.

To create an alarm, perform the following step.

Enter the following command:

```
euwatch-put-metric-alarm [alarm_name] --namespace "AWS/EC2"
-- dimensions "InstanceId=[instance_id]" --statistic [statistic] --metric-name

[metric] --comparison-operator [operator] --threshold [value] --period
[seconds] --evaluation-periods [value] -- alarm-actions [action]
```

For example, the following triggers an Auto Scaling policy if the average CPUUtilization is less than 10 percent over a 24 hour period.

```
euwatch-put-metric-alarm test-Alarm --namespace "AWS/EC2" -- dimensions
"InstanceId=i-abc123" --statistic Average --metric-name CPUUtilization
--comparison-operator LessThanThreshold --threshold 10 --period 86400
--evaluation-periods 4 -- alarm-actions
arn:aws:autoscaling::429942273585:scalingPolicy:
12ad560b-58b2-4051-a6d3-80e53b674de4:autoScalingGroupName/testgroup01:
policyName/testgroup01-pol01
```

Test an Alarm

You can test the CloudWatch alarms by temporarily changing the state of your alarm to "ALARM" using the command: `euwatch-set-alarm-state`.

```
euwatch-set-alarm-state --alarm-name TestAlarm --state ALARM
```

Delete an Alarm

To delete an alarm, perform the following step.

- Enter the following command:

```
euwatch-delete-alarms [alarm_name]
```

For example, to delete an alarm named `TestAlarm` enter:

```
euwatch-delete-alarms TestAlarm
```

Glossary

cluster
A group of resources that contains a CC, an SC and, optionally, a Broker.

availability zone
An availability zone for AWS denotes a large subset of their cloud environment. Eucalyptus refines this definition to denote a subset of the cloud that shares a local area network. Each availability zone has its own cluster controller and storage controller.

AWS
Amazon Web Services

bucket storage
A storage container that accepts objects via PUT and GET commands.

bundling
A virtual machine image splits the image into multiple image parts to facilitate ease of uploading. It also generates an XML manifest file containing metadata referencing the image, including image parts and kernel, which is used to assemble instances of the image.

Cloud Controller
The Cloud Controller (CLC) is the entry-point into the cloud for administrators, developers, project managers, and end-users. The CLC queries the node managers [SM1] for information about resources, makes high-level scheduling decisions, and makes requests to the Cluster Controllers (CCs). As the interface to the management platform, the CLC is responsible for exposing and managing the underlying virtualized resources (servers, network, and storage). You can access the CLC through Amazon's Elastic Compute Cloud (EC2) and through a web-based Eucalyptus Administrator Console.

Cluster Controller
The Cluster Controller (CC) generally executes on a machine that has network connectivity to both the machines running the Node Controller (NC) and to the machine running the CLC. CCs gather information about a set of node machines and schedules virtual machine (VM) execution on specific nodes. The CC also manages the virtual machine networks and participates in the enforcement of SLAs[SM3] as directed by the CLC. All NCs associated with a single CC must be in the same broadcast domain (Ethernet).

dynamic block volume
A dynamic block volume is similar to a raw block storage device that can be used with VM instances. You can create, attach, detach, describe, bundle, and delete volumes. You can also create and delete snapshots of volumes and create new volumes from snapshots

elastic IP
Public IP addresses that you can reserve and dynamically associate with VM instances.

instance type
An instance type defines what hardware the instance has, including the amount of memory, disk space, and CPU power.

kernel/ramdisk pair
A ramdisk contains drivers that direct the kernel to launch appropriate system files when instantiating a virtual machine.

Node Controller
The Node Controller (NC) executes on any machine that hosts VM instances. The NC controls VM activities, including the execution, inspection, and termination of VM instances. It also fetches and maintains a local cache of instance images, and it queries and controls the system software (host OS and the hypervisor) in response to queries and control requests from the CC. The NC is also responsible for the management of the virtual network endpoint.

Storage Controller
The Storage Controller (SC) provides functionality similar to the [5] Amazon Elastic Block Storage (EBS) and is capable of interfacing with various storage systems (NFS, iSCSI, SAN devices, etc.). Elastic block storage exports storage volumes that can be attached by a VM and mounted or accessed as a raw block device. EBS volumes persist past VM termination and are commonly used to store persistent data. An EBS volume cannot be shared between VMs and can only be accessed within the same availability zone in which the VM is running. Users can create snapshots from EBS volumes. Snapshots are stored in Walrus and made available across availability zones. Eucalyptus with SAN support lets you use your enterprise-grade SAN devices to host EBS storage within a Eucalyptus cloud.

VMware Broker

VMware Broker (Broker or VB) is an optional Eucalyptus component activated only in versions of Eucalyptus with VMware support. VMware Broker enables Eucalyptus to deploy VMs on VMware infrastructure elements and mediates all interactions between the Cluster Controller (CC) and VMware hypervisors (ESX/ESXi) either directly or through VMware vCenter.

Walrus

Walrus allows users to store persistent data, organized as buckets and objects. You can use Walrus to create, delete, and list buckets, or to put, get, and delete objects, or to set access control policies. Walrus is interface compatible with Amazon's Simple Storage Service (S3), providing a mechanism for storing and accessing virtual machine images and user data. Note that Walrus access is global to the entire Eucalyptus cloud. This means that it can be accessed by end-users, whether the user is running a client from outside the cloud or from a virtual machine instance running inside the cloud.

Index