Eucalyptus 3.2: Design, Build, Manage

Contents

Eucalyptus History

Eucalyptus was born as a University project of the Computer Science Department at the University of California, Santa Barbara in 2007. The name Eucalyptus is an acronym and stands for **E**lastic **U**tility **C**omputing **A**rchitecture for **L**inking **Y**our **P**rograms **T**o **U**seful **S**ystems. In 2009, the Eucalyptus team started a company (Eucalyptus Systems Inc.) to commercialize Eucalyptus.

Eucalyptus provides is users with a specific set of benefits. Eucalyptus is:

- AWS compatible
- Enterprise-ready
- Open source
- Agile
- Robust and dependable

Eucalyptus Cloud Characteristics

Eucalyptus clouds benefit from a number of key characteristics, including:

- The open source nature of the product
- Adherence to the industry-standard Amazon Web Services API
- The ability to work with multiple hypervisors
- A modular, distributed, and scalable architecture

Open Source

Eucalyptus is open source: if you want to download it, modify it, contribute back to it, or assess its security. Whatever your use case, you have the source code at your fingertips.

The source code is available on GitHub at *https://github.com/eucalyptus*. Bug reports are created and tracked using Jira. Jira can be found at *https://eucalyptus.atlassian.net*.

The Eucalyptus development process is in the open, as are bug reports, community contributions, and security advisories. This means that companies that deploy Eucalyptus can be assured that the product has been thoroughly vetted by the open source community, and any issues discovered can be mitigated in a rapid fashion.

Amazon Web Services Compatible

Amazon Web Services (AWS) is by far the most widely used public cloud in the world. Eucalyptus is highly-compatible with Amazon's Elastic Compute Cloud (EC2), Simple Storage Service (S3), and Identity and Access Management (IAM) APIs. Eucalyptus also provides the Eucalyptus Block Storage (EBS) which is very similar to the Amazon Elastic Block Store (EBS). Eucalyptus regularly examines the need to support additional Amazon APIs. Check the Eucalyptus roadmap at *http://www.eucalyptus.com/eucalyptus-cloud/iaas/roadmap* for plans to add additional API support.

This means that you can reuse your existing AWS-compatible tools and scripts to manage your own private cloud, often with little or no modification. Eucalyptus also provides other functionality similar to AWS, including Amazon Machine Images (called Eucalyptus Machine Images), Availability Zones, and Elastic IP addresses.

Hypervisor Agnostic

Eucalyptus is designed to easily work with several available hypervisors. As of this writing, Eucalyptus IaaS offically supports the KVM hypervisor running on Redhat Enterprise Linux (RHEL) 6 and CentOS 6 operating systems. Additionally, Eucalyptus offers a subscription plug-in that enables support for proprietary VMware ESX/ESXi hypervisor in versions 4.0, 4.1 and 5.0. The Xen hypervisor will work, but has not been fully quality tested by Eucalyptus. Eucalyptus does not support two different hypervisors in the same cloud.

Note: Eucalyptus knows that if you happen to run different hypervisors in different clusters in the same cloud, you must carefully choose images that will run on multiple hypervisors. There are differences when running on Xen versus KVM for example. It can work, but Eucalyptus does not test, recommend, or support it in production.

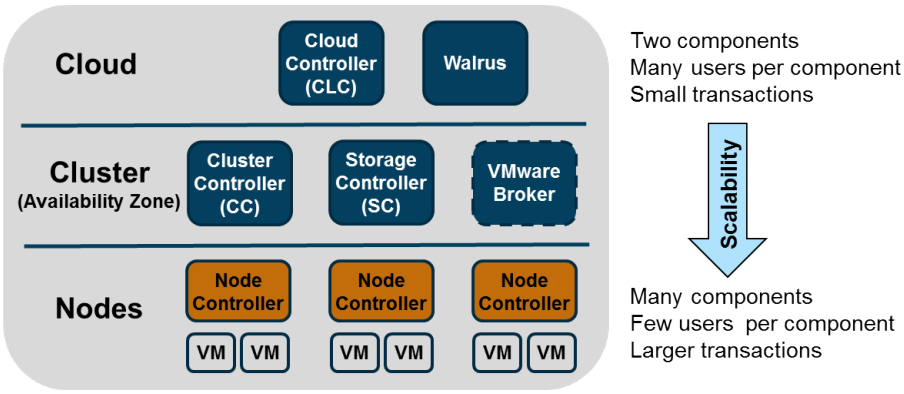
Access to, and management of, hypervisors is controlled by the Eucalyptus software. The underlying hypervisor is abstracted from the user, so they do not have to know which hypervisor platform is in use at any given time. This reduces the need to understand the vCenter Server utility, or command-line utilities like the `vicfg-*`, `xm`, `virsh`, `virt-install`, or others.

Eucalyptus Components and Architecture

Eucalyptus is a modular, distributed, and thus highly scalable cloud computing architecture composed of six distinct components that can be deployed in various architectures.

Eucalyptus Architecture

Eucalyptus' design is modular. In total, Eucalyptus is comprised of six distinct components that are arranged in three layers. These components are software services and can be installed on the same physical server or on separate physical servers as business, security, and resource needs dictate. Components can be installed strategically close to the needed resources. For example, storage services can be installed close to physical storage resources, while management services can be installed close to the resources they manage.



The Eucalyptus components have well-defined communication interfaces via the Web Services Description Language (WSDL). WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either *document oriented* or *procedure oriented* information. Because communication between the components happens in a well-defined but abstract manner, components can be changed or replaced with minimal to no configuration required on the part of other components in the system.

Eucalyptus architecture is very scalable because of its tree-like structure. The top layer contains just two components. These components control all other cloud components. While there are only two components, the transactions they process are small.

While not shown in the illustration above, there can be multiple sets of cluster and node-layer components (up to eight sets). A set of these components is called a cluster. The bottom of the tree has many components but each component only supports a subset of the cloud users. The transactions that are processed at this level are typically larger.

This modularity allows Eucalyptus to be extremely scalable and to achieve optimal performance in diverse environments. It can be installed on a very minimal setup - a single-server test cloud, for example - or installed on dozens of hosts and terabytes of storage.

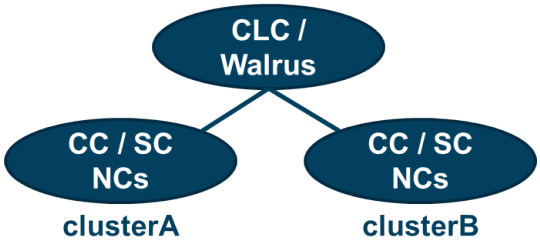
The VMware Broker component is optional and only required if you intend to manage VMware hypervisors. This component is only available with the purchase of a Eucalyptus subscription.

Clusters

An cluster is a subset of the cloud (typically a collection of servers and storage) that shares a local area network and are in the same network broadcast domain. A cluster offers a fixed amount of resources to the cloud, and access to those resources can be controlled via quotas and access control lists. It is a best practice to configure a cluster with server hardware that has the same, or very similar, performance capabilities. This ensures that the cluster offers the same performance capabilities to all instances that are run in the cluster.

A cluster is the Eucalyptus implementation of an Amazon Availability Zone. A cluster could be a single point of failure. Configuring multiple clusters and deploying an application across them increases the availability of the application.

A cluster is also sometimes referred to as a partition. Many Eucalyptus users use the terms cluster, availability zone, and partition interchangeably.

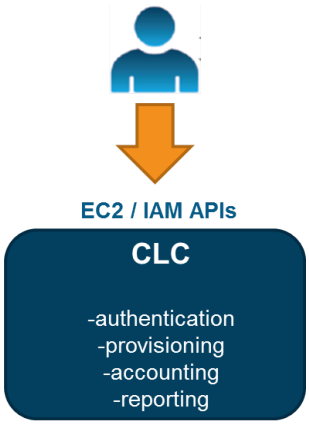


Multiple clusters can be deployed for administrative or technical reasons.

Administrative separations happen due to resource ownership (for example, the Engineering group owns a collection of servers, therefore they get exclusive access to them in the cloud) or compliance purposes (when certain data is legally not allowed to exist in certain places).

Technical separations happen when you want, for example, to provide different service level agreements (SLAs) to different users in the same cloud. Different clusters can be deployed with hardware with different performance capabilities. As such, the SLAs offered to instances running in different clusters can vary.

Cloud Controller

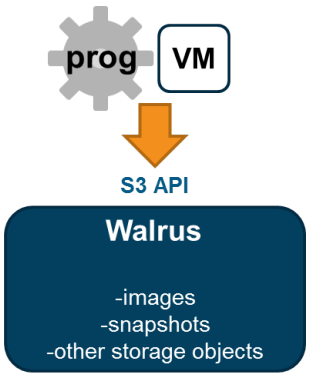


The Cloud Controller (CLC) is a Java-based AWS EC2-compatible interface to a Eucalyptus cloud that offers SOAP and "Query" interfaces, as well as a Web interface to the outside world. It handles all authentication, as well as high-level provisioning, resource scheduling, accounting, reporting, and quota management in the cloud. It also exposes an administrative interface for managing accounts, users, groups, access control, quotas, and vmtype settings.

The administrative interface is accessed at the URL https://<CLC_public_IP>:8443 while other user-level services are accessed at the URL http://<CLC_public_IP>:8773.

Only one active Cloud Controller can be provisioned per cloud, but a second, passive Cloud Controller can be provisioned for high availability purposes.

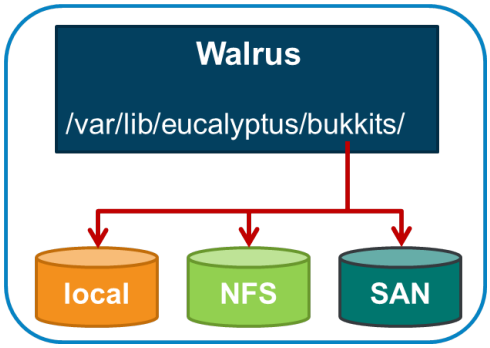
Walrus



The Walrus is a Java-based equivalent to Amazon's Simple Storage Service (S3). The Walrus is the Eucalyptus cloud component that handles bucket-based object storage for the cloud. Walrus provides persistent storage to both instances running within the cloud as well as to programs running outside of the cloud. Programs outside of the cloud can access the Walrus because it exposes S3-compatible SOAP and REST interfaces outside the cloud. Some customers have used the Walrus as a separate Storage-as-a-Service solution. The Walrus also provides storage for the Eucalyptus Machine Images, Eucalyptus Kernel Images, and Eucalyptus Ramdisk Images along with Eucalyptus Block Store volume snapshots.

Only one active Walrus can be configured per Eucalyptus cloud, but a second, passive Walrus can be configured for high availability purposes.

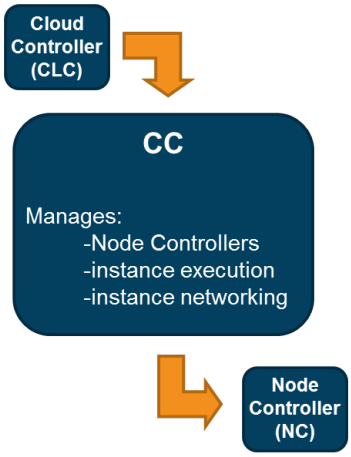
Walrus Storage Types



A variety of storage types can be used to provide the storage space for Walrus buckets. The storage space from this storage should be reachable by the Walrus beneath the directory `/var/lib/eucalyptus/bukkits`. Whichever storage type you choose, consider reliability, disaster recovery, bandwidth, and the ability to increase the capacity of the storage to accommodate future growth.

If you have mounted your Walrus storage over NFS, then ensure that you use the NFS options *hard,nointr,sync*. This ensures that data written to the Walrus is safely written to the back-end storage.

Cluster Controller



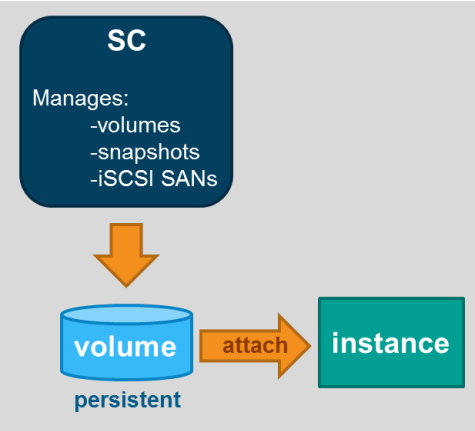
The Cluster Controller (CC) is the front end for a cluster in a Eucalyptus cloud. It is written in C and deployed as a Web service inside Apache. It manages the compute nodes (Node Controllers) in the cluster as well as instance execution.

The Cluster Controller communicates with the Storage Controller and Node Controllers using SOAP with WS-Security. It provides information to the Cloud Controller regarding aggregate resource availability of the Node Controllers, and schedules virtual machine execution on specific Node Controllers.

A Eucalyptus cloud may contain several clusters, however each cluster can have only one active Cluster Controller. A second, passive Cluster Controller may be provisioned for high availability purposes.

Depending on which network mode the cloud is configured in, the Cluster Controller might also perform a number of network functions on behalf of running instances. It might map the private, cloud-internal IP addresses used by instances to their public IP addresses that are used for cloud-external communications. It might also act as an IP router between instances running in the cloud and external nodes. Lastly, it might implement and enforce the firewall rules that comprise security groups.

Storage Controller



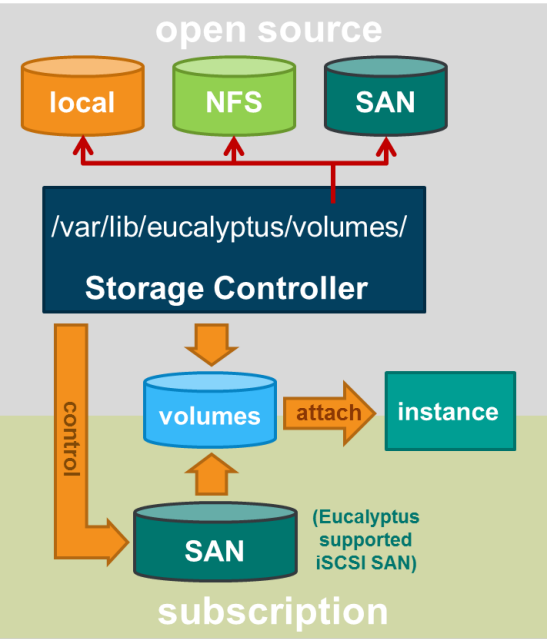
The Storage Controller is a Java-based component that manages dynamic block volumes and snapshots, and is similar to AWS Elastic Block Stores (EBS). It communicates with its Cluster Controller and Node Controllers using SOAP with WS-Security.

The Storage Controller builds and makes Eucalyptus Block Store (EBS) volumes available for attachment to running instances. A volume is a virtual disk and provides instances with access to persistent storage. A volume created in one cluster is not available to instances running in another cluster.

A Eucalyptus cloud may contain several clusters, however each cluster can have only one active Storage Controller. A second, passive Storage Controller may be provisioned for high availability purposes.

If you have purchased Eucalyptus IaaS Subscription then you will have access to add-on software packages that allow the Storage Controller to control a supported iSCSI SAN array. The Storage Controller directs the array to build volumes and make them available for attachment by running instances.

Storage Types



A variety of storage types can be used to provide the storage space for volumes. You can divide these storage types into two broad categories.

First, there are Eucalyptus-supported iSCSI SAN arrays. At the time of writing these include the Dell EqualLogic PS4000 series or PS6000 series arrays, the NetApp FAS2000 series or FAS6000 series arrays, and the EMC VNX series arrays. These arrays directly provide volumes to the Node Controllers, but under the control of the Storage Controllers.

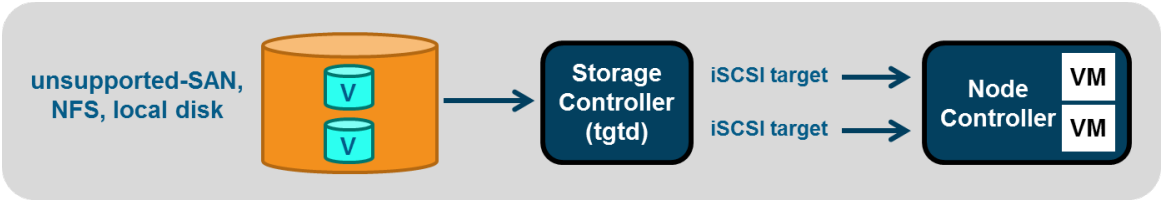
Second, a large variety of other storage types can also be used, but these types do not directly provide volumes to the Node Controllers. These types provide storage space to the Storage Controller which in turn provides the volumes to the Node Controllers. The storage space on these types of storage is accessed through the directory `/var/lib/eucalyptus/volumes` on the Storage Controller.

Eucalyptus does not recommend the use of NFS storage for volumes in production environments due to potential performance issues. If you have mounted storage to your Storage Controller over NFS, then ensure that you use the NFS options *hard,nointr,sync*. Without these options, any Boot-from-EBS instances might become unstable at any time, but especially during periods of heavy storage I/O load.

iSCSI Targets

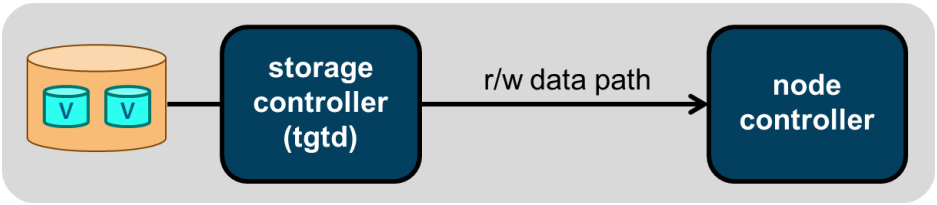
Node Controllers always expect to locate and access volumes as iSCSI targets. If a Eucalyptus-supported iSCSI array is used, the SAN array exports the iSCSI target directly to the Node Controller.

If any other type of storage is used, the storage must appear local to the Storage Controller and the Storage Controller's `tgtd` iSCSI daemon exports the iSCSI target to the Node Controller.

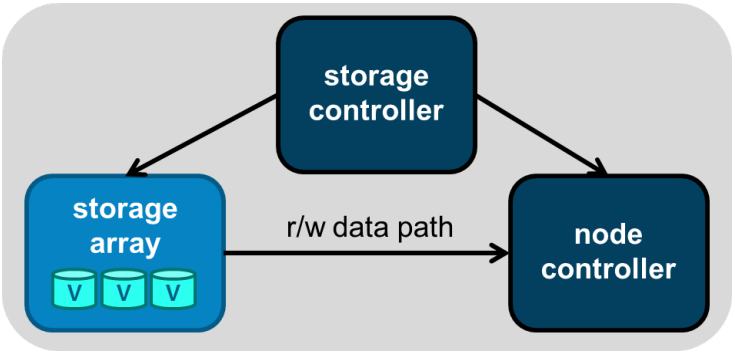


Storage Reliability and Performance

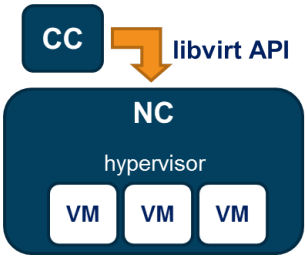
A Storage Controller using local storage is said to be using portable storage and be in portable storage mode. In this mode the Storage Controller is in the storage data path. Depending upon the number of actively used volumes and the type of hardware on the Storage Controller, the Storage Controller could become a storage performance bottleneck. Also, if the Storage Controller fails in a non-Eucalyptus HA environment, access to the volumes will fail along with the instances using those volumes.



With a Eucalyptus-supported iSCSI SAN array, the Storage Controller is not in the data path. In this scenario, it is less likely for the SAN to become a performance bottleneck. Also, because the Storage Controller is not in the data path, a failed Storage Controller will not effect running instances. Even the SAN array itself might be more reliable with the use of built-in RAID functionality and redundant storage array controller hardware.



Node Controller



Node Controllers (NCs) are servers designated for hosting virtual machine instances. The Node Controller software is written in C and deployed as a Web service inside Apache. As of this writing, the two Node Controller hypervisor options in Eucalyptus are KVM and Xen. Only KVM is fully tested and supported by Eucalyptus. Node Controllers communicate with their Cluster Controller and Storage Controller using SOAP with WS-Security.

Node Controllers download and cache Eucalyptus Machine Images, Eucalyptus Kernel Images, and Eucalyptus Ramdisk Images from the Walrus. In response to requests from the Cluster Controller, they also control virtual machine instances' execution, inspection, and termination, as well as query and control the host operating system and hypervisor.

The Cluster Controller uses the libvirt API on the Node Controller to control the hypervisors. Libvirt is implemented on these Node Controllers via the `libvirtd` daemon. The `virsh` shell is also available although it it typically only used for troubleshooting instance problems.

Many Node Controllers can be configured per cluster.

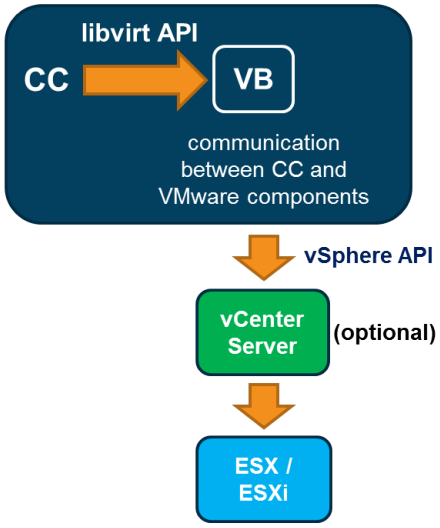
Note: Since Node Controllers run hypervisors, it is possible to manually create virtual machines. This should not be done, except for situations of temporary testing. Eucalyptus is not aware of the resources that manually created virtual machines consume, and will try to launch virtual machine instances on the Node Controller based on the resources of which it is aware. This can cause poor performance or even instance launch failure.

When a new instance is launched, the Node Controller first caches the machine, kernel, and ramdisk image files received from the Walrus directly to its local storage. This increases the speed of subsequent launches of new instances based on these images. The cached image files are located at `/var/lib/eucalyptus/eucalyptus/cache/`. The Node Controller also caches the instances it runs at `/var/lib/eucalyptus/instances/work/<letters_numbers>/i-<nnnnnnnn`.

Note: In the event of a Node Controller failure, instances that were running on it will have to be manually restarted.

If a Node Controller is rebooted with running instances, they will be reported in a *running* state but will not be accessible. These instances will need to be terminated and re-launched. If you cannot terminate them, it might be necessary to stop and start the Cloud Controller and Cluster Controller.

VMware Broker



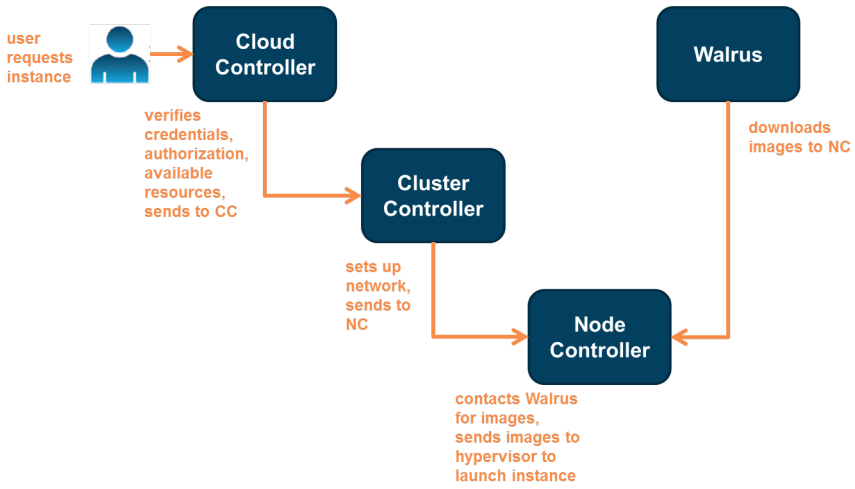
The VMware Broker, available as part of the Eucalyptus IaaS Subscription, overlays existing ESX or ESXi hosts and allows you to provide an AWS-compatible interface to your VMware environment. The VMware Broker mediates all interactions between the Cluster Controller and VMware components, and from the perspective of the Cluster Controller, performs many of the functions that the Node Controller performs in a KVM environment.

The VMware Broker runs on the same physical machine as the Cluster Controller.

The VMware Broker can connect directly to individual ESX or ESXi hosts, or connect to a VMware vCenter Server and manage the VMware hosts through vCenter APIs.

There is a maximum of one active VMware Broker per cluster in a Eucalyptus cloud, but a second, passive VMware Broker can be configured for high availability purposes.

Cloud Operation - Example



To use cloud resources, the user must download cloud credentials from the Cloud Controller. The user uses these credentials to authenticate to the Cloud Controller and submit requests to the cloud. These requests include many types of cloud operations. For example, they could be a request to launch an instance, create a volume, or allocate an elastic IP address.

The Cloud Controller will then determine, based on access controls and quotas, whether the user is authorized to make the request. If they are, then the Cloud Controller will begin to fulfill the request. In the example above, the user is requesting that an instance be launched in the cloud.

When launching an instance, the Cloud Controller will either pick a cluster based on current load, available resources, or user access controls, or will choose a cluster because the user has explicitly requested a specific cluster. Once a cluster has been selected, the Cloud Controller will send the request to launch an instance to the Cluster Controller.

The Cluster Controller will configure the network environment for the instance. The network configuration depends on which network mode was configured for the cloud. This might include reserving and assigning public or private IP addresses, setting up the firewall rules, and configuring a VLAN. The Cluster Controller will then pick a Node Controller on which to run the instance.

The chosen Node Controller will check its local cache to determine if the correct Eucalyptus Machine, Kernel, and Ramdisk images are available. If not, the Node Controller will download these images from the Walrus. Once the necessary images have been downloaded from the Walrus, they are cached on the Node Controller. This eliminates to need to download images again in order to launch another instance based on these images. This means that other instances - based on the same images - should launch more quickly.

Once the Node Controller has the necessary images, it will hand them to the hypervisor for instantiation as a virtual machine. If the hypervisor is handed the images, then the assumption is that the cloud software is working correctly. If the virtual machine fails to boot from the images, it is best to start looking for problems in the images and not the cloud configuration.

Eucalyptus Architecture Examples

Eucalyptus cloud architectures can be thought of in terms of scale and technological diversity. They range from single-server proof-of-concept (POC) environments all the way to multiple-cluster architectures running dozens of Node Controllers and high availability configurations. Six different architecture examples are provided in this section. These six examples can be divided into four categories.

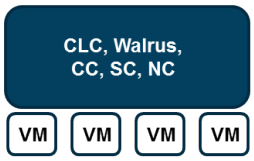
- Proof-of-Concept (POC)
- Single-Cluster
- Multi-Cluster

- Eucalyptus HA

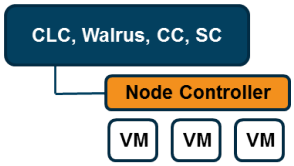
Proof-of-Concept Architectures

A Eucalyptus proof-of-concept (POC) can be created with a simple configuration of one or two hosts.

A single-host architecture is a common architecture used in pre-sale proof-of-concept deployments for testing and evaluation. In this architecture all Eucalyptus software components are installed on a single host. This type of installation is useful for functional testing but is not useful for scale and load testing. This architecture can be easily deployed using the Eucalyptus FastStart ISO or by using a normal software package installation. All storage is typically provided by the local disks on this host.

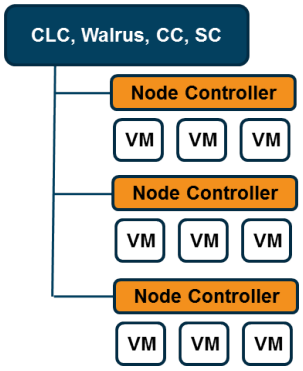


In a two-host proof-of-concept configuration, all front-end components (Cloud Controller, Walrus, Cluster Controller, and Storage Controller) run on one of the hosts, and the other host is configured as a Node Controller. The Walrus and Storage Controller use the internal disk space of the front-end machine. This architecture can also be easily deployed using the Eucalyptus FastStart ISO or by using a normal software package installation.



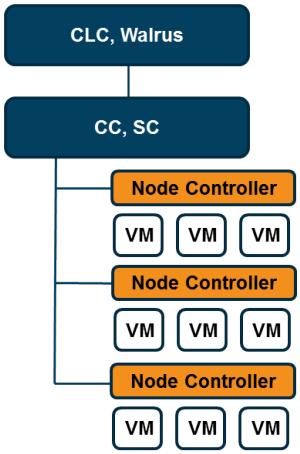
Single Cluster Architectures

Small-to-medium Eucalyptus deployments are usually configured in a single-cluster architecture. Small single-cluster architectures are usually what is deployed during proof-of-concept testing for scale and load, but these deployments are also sufficient for a small business to use in a production environment.

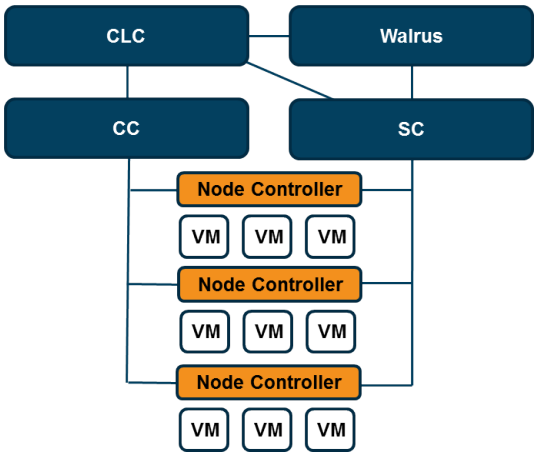


As the number of Node Controllers in a cluster expands, resource demands on the front-end machine may require the separation of the Eucalyptus components in order to maintain optimum performance. In this situation, Cloud

Controller, Walrus, Cluster Controller, and Storage Controller can be separated into various combinations. In some cases, it makes sense to separate the cloud components from the cluster components. In other cases, it may make sense to separate just the Storage Controller, or just the Cluster Controller, from the rest of the components. It depends on the types of resource demands being placed on the components, and the resources available on the physical servers on which they reside.

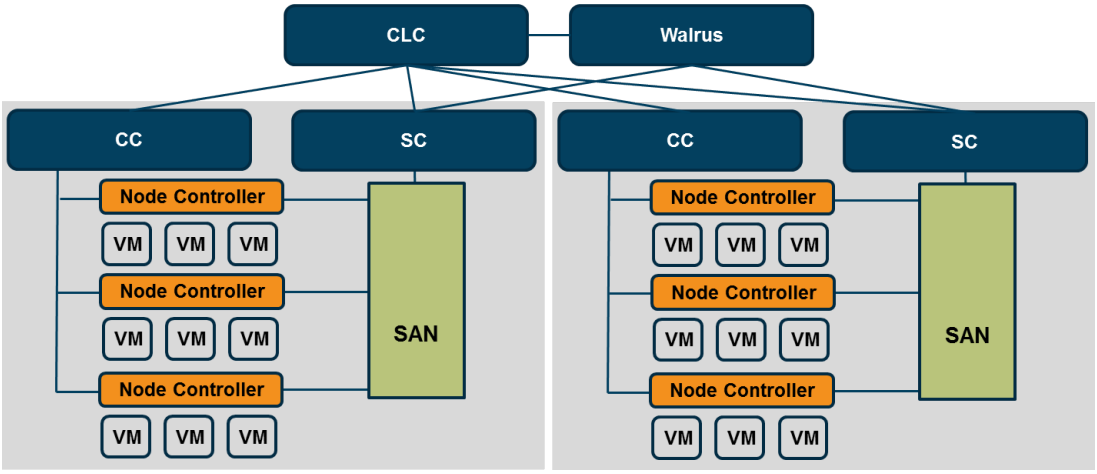


In large single-cluster architectures, it may be useful to separate all of the front-end components to separate physical servers.



Multiple Cluster Architectures

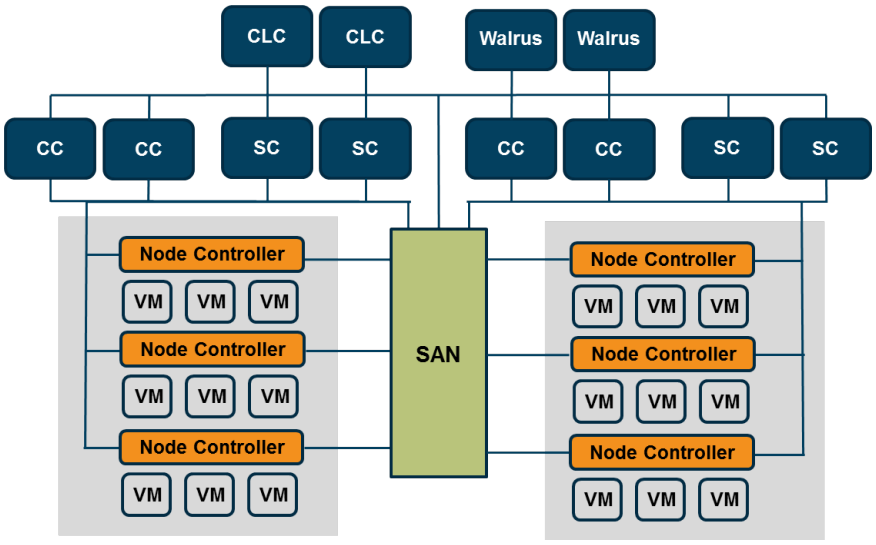
Medium-to-large and complex Eucalyptus deployments may contain up to eight clusters. Companies that need to organize their users according to different Quality of Service (QoS) or Service Level Agreement (SLA) requirements, or companies that have strict policies regarding how their various organizations access resources may find multiple-cluster deployments are the best way to meet the various needs of their users.



Note: Eucalyptus 3 supports up to eight (8) clusters per cloud deployment.

High Availability Architectures

Eucalyptus High Availability (HA) provides redundancy for the cloud and cluster-layer components in the architecture. It allows the configuration of passive services for these components on separate hardware to avoid single-points-of-failure for cloud services. Eucalyptus HA is discussed more extensively in another training module.



Other Information Resources

There are several sources for additional information about Eucalptus.

PEOPLE & NEWS

Team *http://www.eucalyptus.com/about/team*

CEO's blog *http://www.eucalyptus.com/blogs/marten*

CEO's tweets *http://twitter.com/#!/martenmickos*

Press releases *http://www.eucalyptus.com/news?tid=17*

Media coverage *http://www.eucalyptus.com/news?tid=6*

Guest blogs *http://www.eucalyptus.com/news?tid=22*

Interviews *http://www.eucalyptus.com/news?tid=23*

ECOSYSTEM

Partner list *http://www.eucalyptus.com/partners*

Open source community *http://open.eucalyptus.com/*

Planet Eucalyptus *http://planet.eucalyptus.com/*

Business Customers *http://www.eucalyptus.com/about/customers*

Case studies *http://www.eucalyptus.com/about/customers/case-studies*

TECHNOLOGY AND SERVICES

Product description *http://www.eucalyptus.com/products/eee*

Information resources *http://www.eucalyptus.com/resources/overview*

Brief videos *http://www.eucalyptus.com/video*

White papers *http://www.eucalyptus.com/resources/whitepapers*

Services *http://www.eucalyptus.com/services/overview*

Eucalyptus Networking

Eucalyptus offers several different network configurations. This network configuration flexibility is designed to allow Eucalyptus to run in a variety of datacenters with different network requirements and constraints. But with this flexibility comes a certain degree of complexity. This section will introduce several topics and concepts, including the different network modes available in Eucalyptus, IP address ranges you will need to plan for and configure, and instance security through network firewalls and VLAN configuration. It will also cover how to edit the `eucalyptus.conf` file and make the changes necessary to configure the different network modes.

Network Modes

Eucalyptus cloud networking supports four network modes. The four modes include:

- SYSTEM (default)
- STATIC
- MANAGED
- MANAGED-NOVLAN

SYSTEM and STATIC modes are very similar, the main difference being the location of the DHCP server that provides IP addresses to instances as they boot up. MANAGED and MANAGED-NOVLAN are very similar, the main difference being support for placing groups of instances in separate VLANs for additional network security.

The network mode is configured by modifying the VNET_MODE parameter `/etc/eucalyptus/eucalyptus.conf` file on the Cluster Controller and Node Controller hosts.

The choice of network mode depends on two factors:

- Which Eucalyptus network features do you require or desire?
- How much control do you have over the underlying physical network?

Security groups, elastic IP addresses, and VLAN network isolation are only available in certain network modes. If you require or desire one or more of these features then you would have to configure the network mode that supports these features. For example, MANAGED mode supports all three of these features.

In MANAGED mode, Eucalyptus will automatically create and destroy VLANs, as needed, when instances are launched and terminated in different security groups. This means that certain segments of the underlying physical network infrastructure should not already have VLANs configured as this could interfere with normal Eucalyptus operation. If this is not possible to configure, then MANAGED mode should not be used. More detail about VLAN configuration is provided in a later section of this training module.

SYSTEM Mode

SYSTEM is the default network mode for Eucalyptus clouds. It assumes that virtual machine instances will be assigned IP addresses by an external DHCP server.

Eucalyptus clouds in SYSTEM mode also do not support the use of elastic IP addresses, security groups, or the further isolation instances in different security groups by placing them in separate VLANs. Both security groups and elastic IP addresses are discussed in more detail in other sections of this training module.

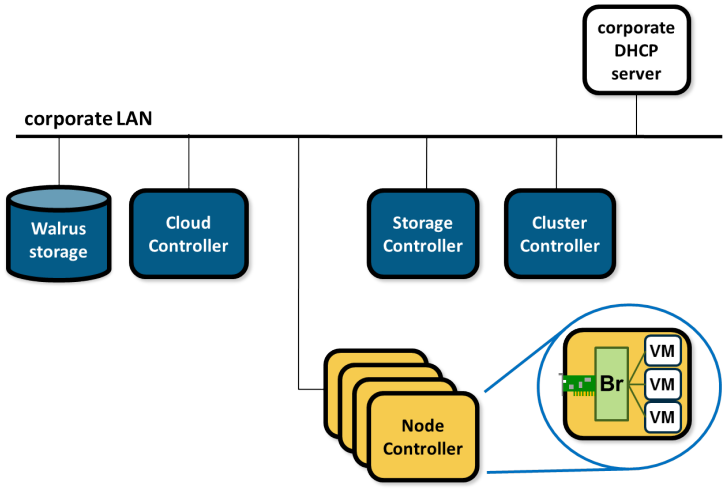
DHCP Server	***External to Eucalyptus***
Elastic IP Addresses[1]	**Not available**
Security Groups[2]	**Not available**
VM Layer 2 Isolation	**Not available**

[1]An elastic IP address is a public IP address that the user reserves and manually assigns to their instance.

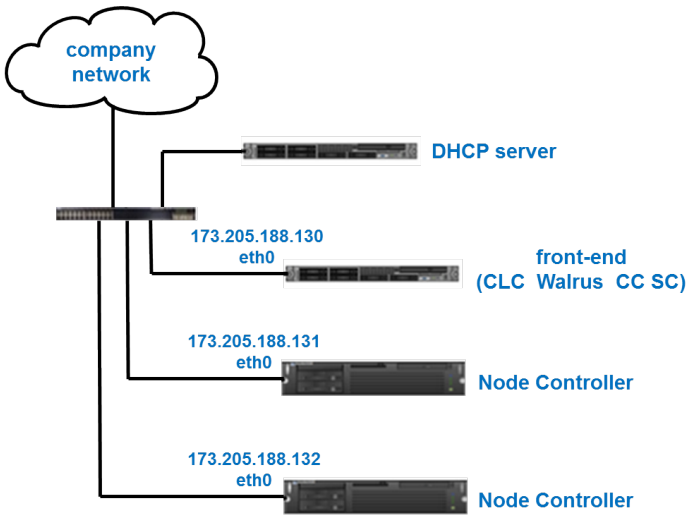
[2]A security group is a set of firewall rules applied to all instances that are members of the security group.

SYSTEM mode is most often used in setting up test or proof-of-concepts (POCs) environments. It is seldom used in production environments. SYSTEM mode is the least intrusive network mode in that it often requires no substantial changes to the existing datacenter network.

Instances receive a single IP address from the organization's DHCP server. All instances have direct network access through the network bridge in the Node Controller, and will thus appear as though they are physical machines on the network. You must ensure that a bridge is configured on each Node Controller prior to installing the Eucalyptus software. It is up to the IT administration team to provide and manage any additional network functionality like network firewalls or network address translation.



The following is an example of what a physical layout might look like for a Eucalyptus cloud configured in SYSTEM mode. In this example, the front-end server, Node Controllers, and all virtual machine instances have IP addresses on the 173.205.188.0/24 network.



The requirements for SYSTEM mode are:

- A pre-existing DHCP server on the network
- A range of IP addresses available for use by Eucalyptus hosts and instances
- The Ethernet interfaces on the Node Controllers that communicates with the Cluster Controller must be bridged before installing the Eucalyptus software.

STATIC Mode

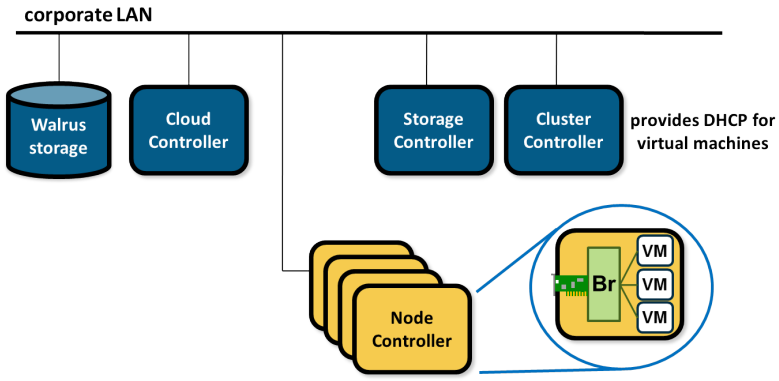
STATIC mode is similar to SYSTEM mode except that Eucalyptus provides a DHCP server that responds to IP address requests from instances as they boot up. Like SYSTEM mode, STATIC mode does not support elastic IP addresses, security groups, or VLAN isolation between instances in different security groups.

DHCP Server	*Cluster Controller*
Elastic IP Addresses	Not available
Security Groups	Not available
VM Layer 2 Isolation	Not available

STATIC network mode assumes that there are no other DHCP servers on the network or that those DHCP servers are not able to respond to IP address requests sent from instances. The Eucalyptus Cluster Controller runs a DHCP server daemon - `/usr/sbin/dhcpd41` for example - and assumes responsibility for assigning IP addresses to instances using a Eucalyptus generated `/var/run/eucalyptus/net/euca-dhcp.conf` configuration file. Each instance receives a single MAC and IP address and appears as though it were a physical machine on the network. All instances have direct network access through the network bridge configured on the Node Controller.

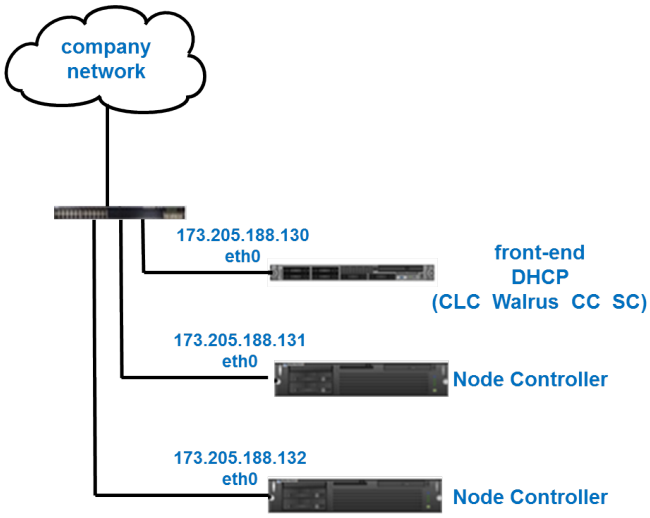
The Eucalyptus administrator is responsible for creating a MAC address-to-IP address mapping list using the VNET_MACMAP parameter in the `eucalyptus.conf` file on the Cluster Controller. When a virtual machine

instance launches, an unused MAC/IP address pair is chosen for the instance. Because of the labor-intensive, human-error-prone nature of STATIC mode, it rarely gets used in large production environments.



The IT administration team is responsible for providing and managing additional network features such as firewalls and network address translation.

Below is an example of a physical layout for STATIC mode networking. In this example, the IT group must somehow isolate nodes on the public network from the DHCP server running on the Cluster Controller. In this example, the front-end server, Node Controllers, and all virtual machine instances have IP addresses on the 173.205.188.0/24 network.



The requirements for STATIC mode are:

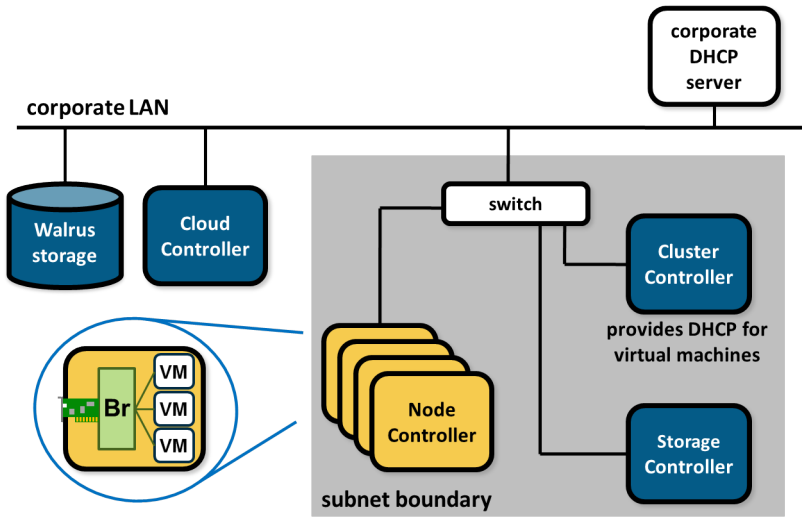
- A range of IP addresses available for use by Eucalyptus instances and components
- No pre-existing DHCP server on the network, or if one exists, it should not respond to Eucalyptus instance requests
- The Ethernet interface on the Node Controllers that communicate with the Cluster Controller must be bridged before installing the Eucalyptus software.

MANAGED and MANAGED-NOVLAN Modes

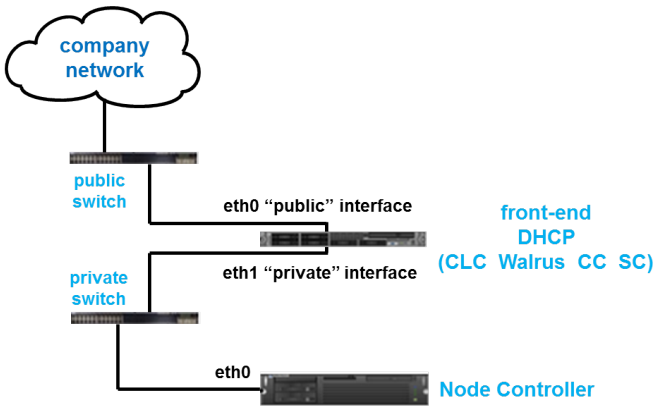
The two MANAGED networking modes are the most common modes used in Eucalyptus cloud deployments. In either mode, the Eucalyptus cloud assumes responsibility for assigning IP addresses to instances in a controlled subnet, regardless of the presence of a DHCP server on the corporate network. In addition, the user can configure and use both elastic IP addresses and security groups. The only difference between MANAGED and MANAGED-NOVLAN is that MANAGED mode utilizes VLAN tagging for virtual machine instance isolation between security groups, whereas MANAGED-NOVLAN does not.

DHCP Server	***Cluster Controller***
Elastic IP Addresses	**Available**
Security Groups	**Available**
VM Layer 2 Isolation	**MANAGED mode only**

Like STATIC and SYSTEM modes, in MANAGED-NOVLAN mode a bridge must be configured on the Node Controllers for Eucalyptus to use. In MANAGED mode, a bridge is automatically created by the Eucalyptus software and therefore should not be created ahead of time. In MANAGED mode, for each security group there is a bridge automatically created on both the Cluster Controller and Node Controller. These bridges are VLAN tagged interfaces.



The physical layout of the MANAGED (-NOVLAN) modes requires isolation of the DHCP server on the Cluster Controller to prevent it from giving out IP addresses to non-Eucalyptus machines. The IT administration team must take external precautions to ensure that this does not happen.



Cluster Controller on front-end host will act as router between VMs and company network

IP Network Operation

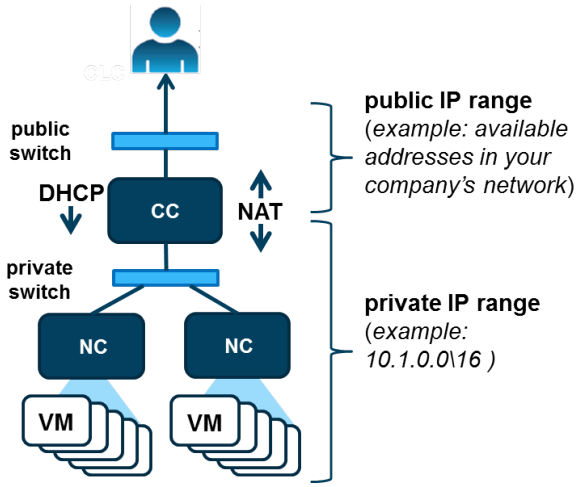
In MANAGED and MANAGED-NOVLAN modes, instances use two IP address ranges. Each instance is assigned:

- A public IP address for communication with hosts external to the cloud
- A private IP address for communication with other instances within the cloud

The public IP address range is configured using the VNET_PUBLICIPS parameter in the `/etc/eucalyptus/eucalyptus.conf` file on the Cluster Controllers. The private IP address range is configured using the VNET_SUBNET and VNET_NETMASK parameters, also on the Cluster Controllers. These parameter settings should be identically configured across all Cluster Controllers in the cloud.

A private IP address is assigned to an instance directly by the Cluster Controller's DHCP server and is displayed inside the instance when using the `ifconfig` or `ipconfig` command. Behind the Cluster Controller, instances only use their assigned private IP addresses.

The public IP address is selected from a pool of public IP addresses that was configured the administrator when the cloud was installed. In front of the Cluster Controller, instances are known by their assigned public IP addresses. The Cluster Controller maps the private IP address to a public IP address via the *nat* table in `iptables`. The Cluster Controller automatically manages the moment-by-moment `iptables` configuration.



Note:

Eucalyptus physical hosts also require IP addresses.

Note:

Remember that this is a private cloud so *public* IP addresses are more than likely IP addresses that are internal to the company but visible to users and applications outside the cloud.

The diagram below is an example of IP address assignment in the Eucalyptus cloud.

The physical Node Controller hosts have been assigned addresses on the 205.16.3.0 subnet. The private Ethernet interface of the Cluster Controller has also been assigned an IP address on the 205.16.3.0 subnet. This allows the Eucalyptus software running on these hosts to communicate.

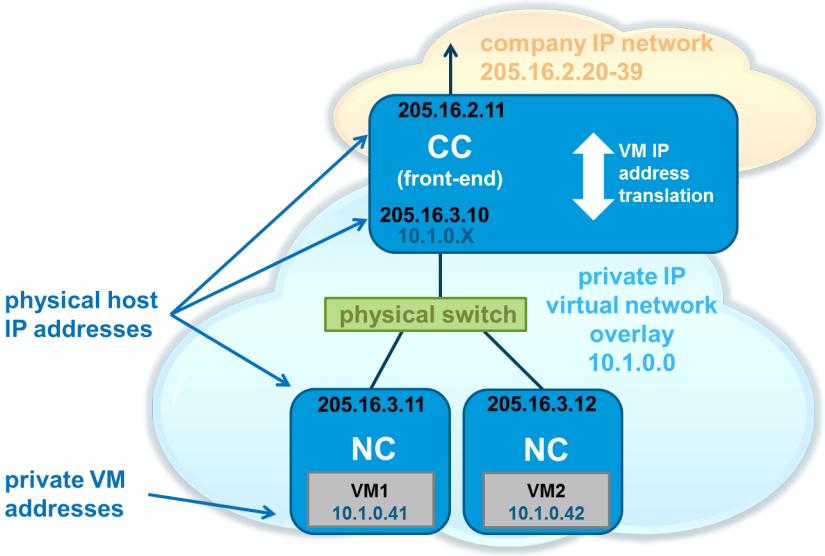
The public Ethernet interface of the Cluster Controller has been assigned an IP address on the 205.16.2.0 subnet. This subnet is behind the company firewall and is populated with other hosts within the company.

A range of private IP addresses has been configured for the virtual machines running inside the cloud. In this illustration, the private IP addresses will all begin with the network prefix of 10.1.0.0. For example, VM1 was assigned the IP address 10.1.0.41 while VM2 was assigned the IP address 10.1.0.42.

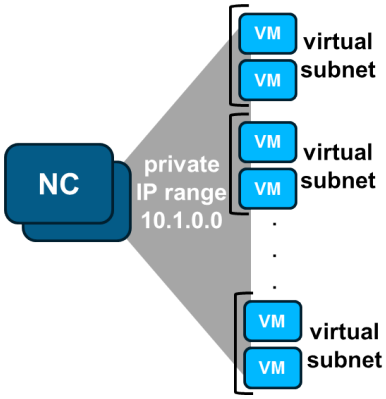
If VM1 needs to communicate with VM2 through the physical switch, it will use its private IP address to communicate with the private IP address of VM2. Because the virtual machines are in the same broadcast domain, they can determine each other's MAC addresses using the ARP protocol and directly deliver Ethernet packets to one another.

A range of company IP addresses has also been reserved for the virtual machines in the cloud. The range in this example is 205.16.2.20-39. As virtual machines are launched in the cloud they are assigned an available company IP address from this range. This allows a virtual machine inside the cloud to communicate with other hosts outside the cloud but inside the company network. In order for this communication to occur, there has to be a way to route an address on the 101.0.0 subnet to the 205.16.2.0 subnet.

This routing is made possible when the Eucalyptus software on the Cluster Controller assigns the private Ethernet interface an address on the 10.1.0.0 network. This address is designated as the router address for the virtual machines running on the Node Controllers. Once a network packet from the virtual machine arrives at the Cluster Controller's private Ethernet interface, the `iptables`' network address translation feature translates the private IP address to one of the available, and reserved, company IP addresses. Once the translation is complete, the Cluster Controller routes the packet to the public Ethernet interface where it is delivered to the 205.16.2.0 subnet. The network address translation and routing process is reversed for packets returning or originating from the company network.



The private IP address range is not a single, large network. The Cluster Controller automatically divides the instances' private IP address range into subnets using administrator-configured parameters in the `/etc/eucalyptus/eucalyptus.conf` file on the Cluster Controllers. The subnet configuration in the `eucalyptus.conf` file determines the maximum number of security groups (one security group is available per configured subnet) and the maximum number of instances per security group (per subnet). This is illustrated below.



Ten IP addresses are reserved by Eucalyptus in every subnet. These IP addresses include the subnet's network and broadcast addresses, along with eight IP addresses that are used as gateway IP addresses for routing IP packets from virtual machines to hosts outside the cloud. Considering these ten reserved IP addresses per subnet, if you configure your cloud for 32 addresses per subnet you can run up to 22 instances (32-10) in each security group.

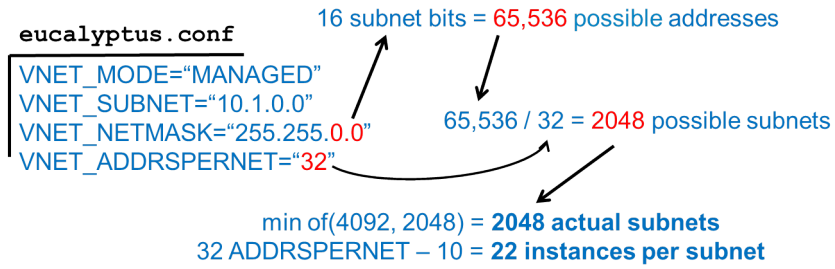
In MANAGED mode when an instance is launched in a security group, that security group is also assigned a VLAN ID. That VLAN ID is reserved for that security group until all instances in the security group have been terminated. There are 4095 possible VLAN IDs, but Eucalyptus only uses VLAN IDs 2-4094 by default.

The main parameters in the `eucalyptus.conf` file that configure instance private IP networking are:

- **VNET_MODE**: determines the network mode, can be SYSTEM, STATIC, MANAGED, MANAGED-NOVLAN
- **VNET_SUBNET**: determines the network prefix used for the instances' private IP addresses
- **VNET_NETMASK**: determines the total address space (range) available to for the instances' private IP addresses
- **VNET_ADDRSPERNET**: determines how the private IP address space will be subnetted by Eucalyptus. A larger number of addresses per subnet will result in few subnets (security groups) while a smaller number of addresses per subnet results in a larger number of subnets (security groups).

These VNET settings must be configured identically across all Cluster Controllers in the cloud.

The illustration below is an example showing how specific VNET settings would affect the number of security groups and instances per security group. Because this is MANAGED mode, VLAN IDs have to be considered as well.

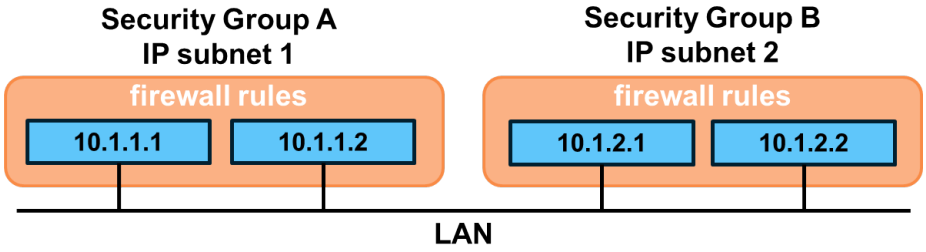


To prevent different Cluster Controllers from allocating the same public or private IP addresses, each Cluster Controller will broadcast a set of *capabilities* # I have this many public IPs, this many private subnets, this networking mode, etc # up the Cloud Controller. When it comes time to start a network or run some instances, the Cloud Controller will send a unique index down to the Cluster Controller which will use the index to select a corresponding actual network value (whether it be public IPs, private IPs, or entire security group subnet ranges). Since the Cloud Controller is the component that knows about multiple clusters, it is the one responsible for keeping those bits uniquely allocated.

Instance Isolation

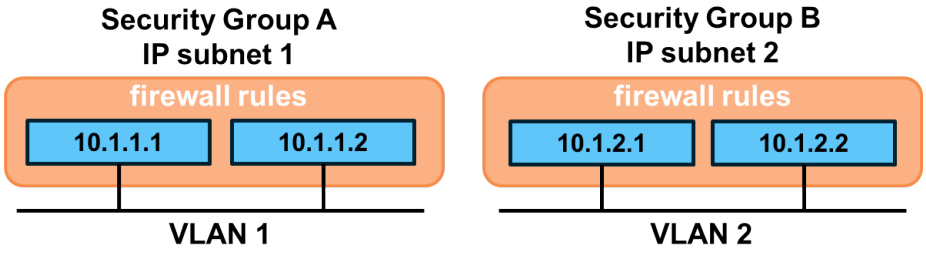
Instance isolation is the one key area of difference between MANAGED and MANAGED-NOVLAN modes.

In MANAGED-NOVLAN mode, instance isolation is managed only at the IP layer through firewalls implemented as part of security groups. While the user decides which network connections are allowed, the actual firewalls are automatically managed by Eucalyptus by manipulating entries in the *filter* table in `iptables`. While placing instances in different security groups separates them into different subnets which can be firewall protected, they are still on a single virtual LAN. Thus in an infrastructure that permits permiscuous network switch operation, an instance in one virtual subnet could run a packet analyzer and see traffic coming from another virtual subnet.



If this is a concern and you need to run in MANAGED-NOVLAN mode, you will need to configure multiple clusters, each with its own security group. Otherwise, you will need to switch to MANAGED mode.

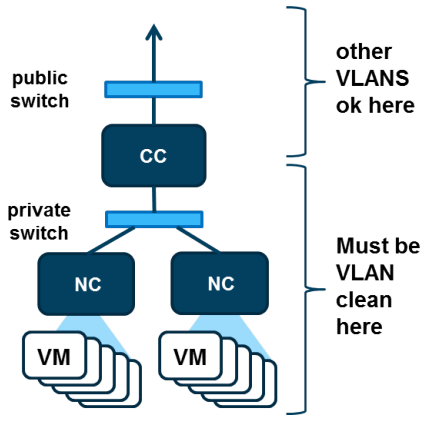
In MANAGED mode, instance isolation is managed by firewalls at the IP layer through security groups (via the `iptables` *filter* table) just like in MANAGED-NOVLAN mode. However in MANAGED mode, each security groups also receives a separate VLAN ID. This prohibits, even in a network infrastructure with permiscuous switches, an instance in one virtual subnet from seeing traffic coming from an instance in another virtual subnet.



In MANAGED mode, the Cloud Controller picks a random, unused VLAN ID number and assigns it to a security group when the first instance in that security group is launched. When all instances in a security group have been terminated, the VLAN ID is returned to the pool of available VLAN IDs. If a security group is reactivated, it may receive a different VLAN ID than it had before.

VLAN-Clean Testing

In MANAGED mode the Cloud Controller will automatically ensure that each security group (subnet) is assigned a unique VLAN ID. This means that Ethernet packets flowing between the Cluster Controller and the Node Controllers will be VLAN tagged. This creates two requirements for the network switches between the Cluster Controller and Node Controllers. First, they must be able to read and forward VLAN-tagged packets. Second, these network switches must not already be using any VLAN IDs that Eucalyptus will be use for security groups. Network switches that can read and forward VLAN-tagged packets and do not have any VLAN ID conflicts are said to be *VLAN clean* from a Eucalyptus perspective.



To test whether the network segment between the Cluster Controller and Node Controller is VLAN clean will require two private IP addresses, a single VLAN ID, and some configuration work on the Cluster Controller and Node Controller network interfaces.

On the Cluster Controller, configure the network interface that is reachable by the Node Controller with virtual network interface. Configure the virtual network interface with a VLAN ID and an IP address. For example:

- `vconfig add eth1 10`
- `ifconfig eth1.10 192.168.1.1 up`

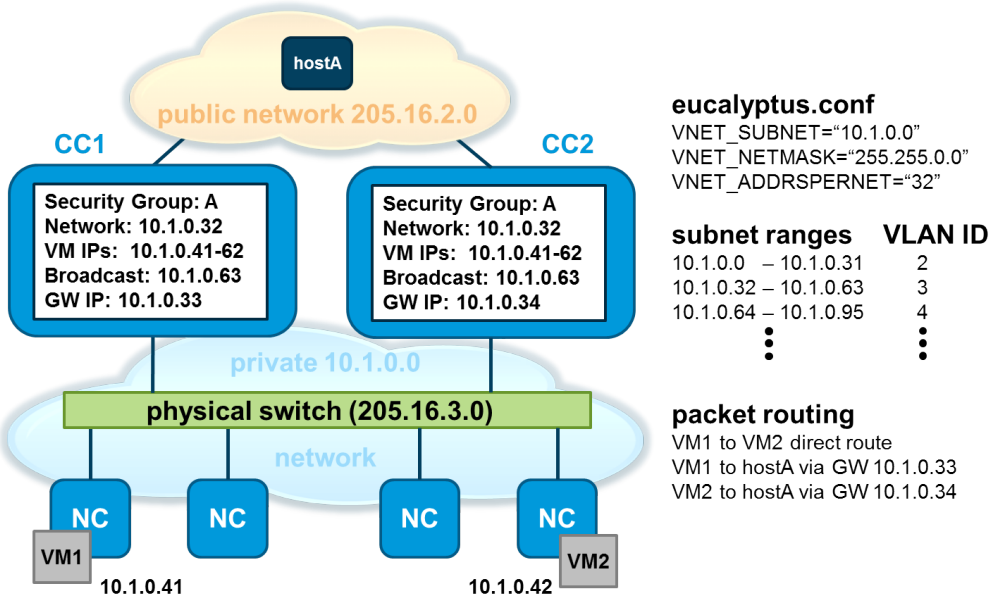
On the Node Controller, configure the network interface that is reachable by the Cluster Controller with a virtual network interface. The virtual network interface should be configured on the same VLAN ID but have a different IP address. For example:

- `vconfig add eth1 10`
- `ifconfig eth0.10 192.168.1.2 up`

Once the network interfaces are configured, on VLAN 10 in this example, trying using the `ping` command from one host to the other host. If it is successful then the network segment is VLAN clean. While this only tests a single VLAN ID, it does indicate that the network infrastructure does support VLAN tagging.

Following the example above, you would remove the virtual interfaces using the syntax `vconfig rem eth1.10` and `vconfig rem eth0.10`.

Routing - Two Cluster Controllers but a Single Subnet



In the example above, each subnet has 32 possible IP addresses. However, ten addresses in each subnet cannot be assigned to virtual machines. In each subnet one IP address is used for the broadcast address, one address is used for the network number, and eight addresses are used as gateway IP addresses for routing IP packets from virtual machines to hosts outside of the cloud.

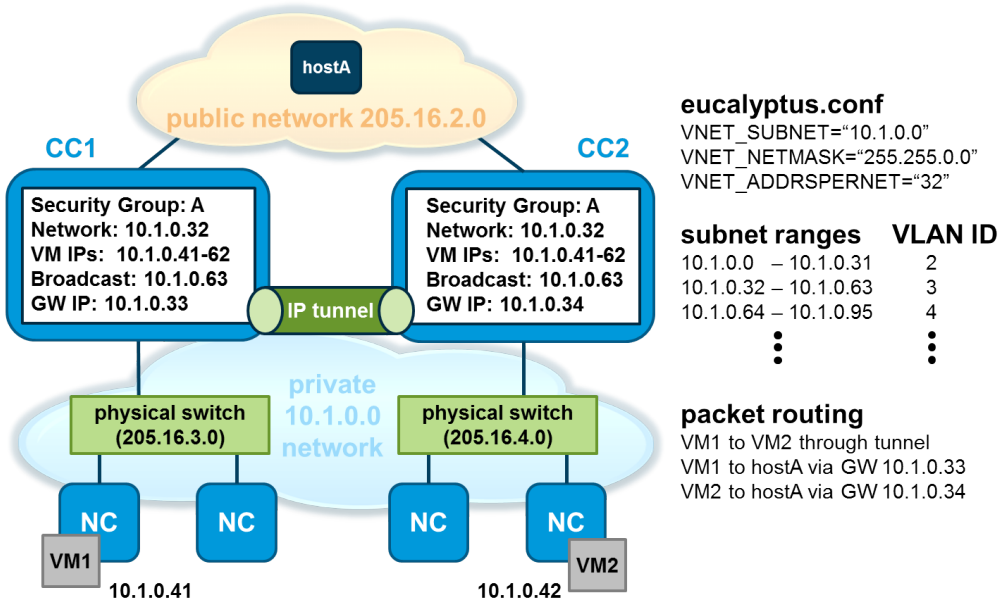
Why are eight IP addresses in each subnet reserved as gateway IP addresses? Because each subnet is also a security group and a security group can span up to eight Cluster Controllers. Each Cluster Controller requires its own gateway IP address for each security group.

Security group A contains virtual machines on two Cluster Controllers so two of the possible eight gateway IP addresses in that subnet are actually in use. VM1 communicates directly with VM2 across the physical switch using the private IP addresses. However, VM1 and VM2 must use the gateway IP addresses on their Cluster Controllers to communicate with hostA.

The Cloud Controller chooses a random VLAN ID and IP address range for the next security group. VLAN IDs 0 and 1 are not used as they reserved.

The Cloud Controller chooses an instance IP address at random from the range of available IP addresses in the subnet.

Routing - Two Cluster Controllers but Multiple Subnets



In the example above, the two Cluster Controllers are not on the same subnet. In this scenario, Eucalyptus must use the vtun interfaces on the Cluster Controllers to create an IP tunnel between the two Cluster Controllers. The vtun interface depends on the two Cluster Controllers having a physical network route to one another. The IP tunnel is created across this network connection. The IP tunnel creates the illusion that VM1 and VM2 in security group A are on the same physical subnet. From the perspective of instances sharing a security group, they can broadcast ARP packets, receive ARP responses, and have a direct route between themselves.

If the routing table on the Cluster Controllers is watched, you would see tunnels being created and destroyed as instances in the same security group are being launched and terminated on the two Cluster Controllers.

Tunnel interfaces are a point of congestion and latency and therefore this configuration is not a Eucalyptus recommended configuration.

MANAGED and MANAGED-NOVLAN Network Mode Requirements

The requirements for the MANAGED and MANAGED-NOVLAN modes are:

- A range of public IP addresses must be available for instances.
- A range of private IP addresses must be available for the instances.
- A set of physical IP addresses must be available for the Eucalyptus components.
- The Cluster Controller firewall must be compatible with the dynamic changes performed by Eucalyptus.
- The network must be VLAN-clean (MANAGED mode only).

Note: The Linux `iptables` firewall in RHEL and CentOS 6 is compatible with Eucalyptus software.

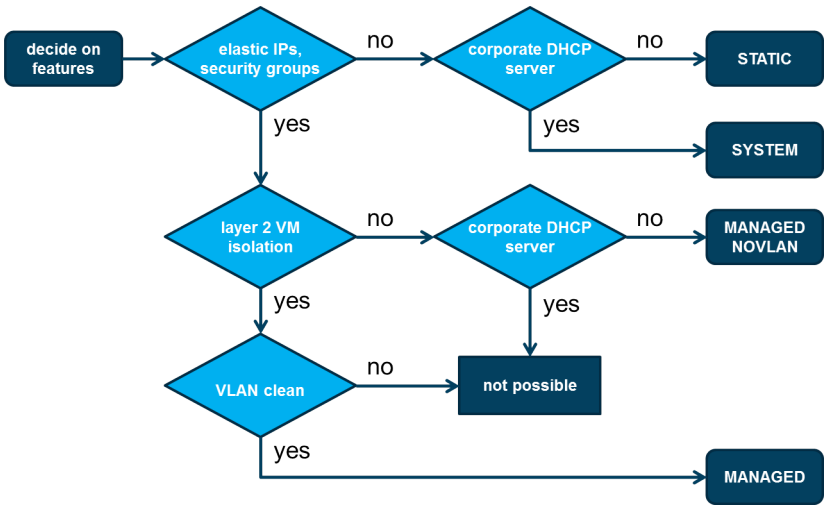
Network Modes Summary

Below is a table which details the different network modes and the options available in each one.

	SYSTEM	STATIC	MANAGED-NOVLAN	MANAGED
DHCP Server	Corporate	Yes	Yes	Yes
Elastic IP Addresses	No	No	Yes	Yes
Security Groups	No	No	Yes	Yes
VM Layer 2 Isolation	No	No	No	Yes

Choose a Network Mode

When choosing a network mode, you can use the following flow chart to help you determine which mode is appropriate for your environment:



The Eucalyptus Configuration File

The Eucalyptus cloud is primarily configured from one central file which is present on all cloud hosts. This file is named `eucalyptus.conf` and is located in the `/etc/eucalyptus` directory.

The file consists of multiple *parameter="value"* entries that provide startup options used by Eucalyptus' initialization scripts. There are parameters to configure the Cloud Controller, Walrus, Cluster Controller, Storage Controller, VMware Broker, and the Node Controller. Parameters that control network configuration must be modified on all Cluster Controllers and Node Controllers.

Note: **CAUTION!** If you edit a networking related value in `eucalyptus.conf` in a configured cloud, you will need to perform a clean restart of all running Cluster Controllers. Use the following procedure:

1. Terminate all instances
2. Run the following command:

```
service eucalyptus-cc cleanrestart
```

The configuration file is often self-explanatory as it contains many comments and samples on how to configure each component. The configuration file available at installation time is sufficient to get a cloud system up and running in SYSTEM network mode. The SYSTEM network mode provides limited functionality and you likely might be required to change to a more appropriate network mode in a production cloud.

If you would like additional information about the network parameters in the `eucalyptus.conf` file, see the *Installation Guide* at *http://www.eucalyptus.com/docs*.

Front-End Network Parameters

These are network parameters that must be set in `eucalyptus.conf` for each network mode on the front-end host(s):

MANAGED	MANAGED-NOVLAN	STATIC	SYSTEM
VNET_MODE	VNET_MODE	VNET_MODE	VNET_MODE
VNET_PUBINTERFACE	VNET_PUBINTERFACE	VNET_PRIVINTERFACE	
VNET_PRIVINTERFACE	VNET_PRIVINTERFACE	VNET_DHCPDAEMON	
VNET_DHCPDAEMON	VNET_DHCPDAEMON	#VNET_DHCPUSER	
#VNET_DHCPUSER	#VNET_DHCPUSER	VNET_SUBNET	
VNET_SUBNET	VNET_SUBNET	VNET_NETMASK	
VNET_NETMASK	VNET_NETMASK	VNET_BROADCAST	
VNET_DNS	VNET_DNS	VNET_ROUTER	
VNET_ADDRSPERNET	VNET_ADDRSPERNET	VNET_DNS	
VNET_PUBLICIPS	VNET_PUBLICIPS	VNET_MACMAP	
#VNET_LOCALIP	#VNET_LOCALIP		

Note: Options listed above that are preceded by a # symbol are optional. Whether or not they must be set to some value depends on the cloud configuration.

If your Cluster Controller and Cloud Controller are running on separate hosts, you might have to configure the VNET_CLOUDIP parameter. You would configure the parameter if the Cloud Controller cannot automatically determine the correct public Ethernet interface. This is typically not the case.

- VNET_CLOUDIP="<ip_of_cloud_controller>"

If your Cluster Controller and Cloud Controller are running on separate hosts, and there are multiple Cluster Controllers, you might have to configure the VNET_LOCALIP parameter. You would configure the parameter if the Cluster Controller cannot automatically determine the correct public Ethernet interface. If this is the case, then you will experience virtual machine routing problems:

- VNET_LOCALIP="<ip_of_cluster_controller>"

Node Controller Network Parameters

These are the network parameters that must be set in eucalyptus.conf on the Node Controller host(s):

MANAGED	MANAGED-NOVLAN	STATIC	SYSTEM
VNET_MODE	VNET_MODE	VNET_MODE	VNET_MODE
VNET_PUBINTERFACE	VNET_PUBINTERFACE	VNET_BRIDGE	VNET_BRIDGE
VNET_PRIVINTERFACE	VNET_PRIVINTERFACE		
VNET_BRIDGE	VNET_BRIDGE		

Eucalyptus IaaS Software

There is only a single version of the Eucalyptus software (starting with version 3.1). Both Eucalyptus IaaS and Eucalyptus IaaS Subscription have the same core cloud software.

The Eucalyptus IaaS software includes:

- Installation packages

The Eucalyptus IaaS Subscription software includes:

- Installation packages
- VMware hypervisor support software
- SAN array support software

The difference between Eucalyptus IaaS and Eucalyptus IaaS Subscription is that with a subscription, users get access to proprietary, add-on software. This add-on software includes SAN drivers for supported SAN arrays and VMware software to support the ESX/ESXi hypervisors. All core Eucalyptus IaaS functionality is included whether you have a subscription or not. Customers always have access to the source code on GitHub at *https://github.com/eucalyptus*.

Eucalyptus Installation Requirements

Eucalyptus has a number of installation requirements. These include:

- Infrastructure host software requirements
- CPU and memory requirements
- Storage requirements
- Network and firewall requirements

Infrastructure Host Software Requirements

Eucalyptus components require physical hosts. Installing Eucalyptus components on virtual machines is not supported.

Eucalyptus software is officially supported on two Linux distributions. These include:

- CentOS 6
- RHEL 6

To install and configure Eucalyptus on these operating systems you will need to have root or sudo access. You must also have Secure Shell (SSH) access to the hosts running Eucalyptus. SSH is used, for example, during the process of registering components with the Cloud Controller or the Cluster Controller. It is also important to ensure that time is synchronized across the hosts running Eucalyptus components. Synchronized time ensures that the Eucalyptus components can properly authenticate to each other and also helps in the event that you might need to troubleshoot an issue by viewing log files across hosts. Using Network Time Protocol (NTP) software for time synchronization is recommended.

While it is usually possible to use Xen and KVM with any of the supported distributions, the level of effort necessary, and the quality of the resulting platform for different distribution-hypervisor combinations varies. Before choosing an open source hypervisor and a Linux distribution, we recommend that you consider the level of support that the community reports for a specific combination. Eucalyptus officially supports the KVM for CentOS 6 and RHEL 6. If you install Eucalyptus from packages on CentOS or RHEL 6, KVM will be automatically installed on the Node Controllers.

For those installing Eucalyptus IaaS Subscription, the add-on software provides support for VMware ESX/ESXi 4.0, 4.1, or 5.0 hosts. You may connect the VMware Broker to each individual ESX or ESXi host, or optionally connect to a vCenter Server 4.0, 4.1, or 5.0 host that manages a collection of ESX or ESXi hosts.

CPU Requirements

Each host requires a minimum of two, 2GHz x86_64 cores. However, most hosts will perform better with more cores. Many hosts running cloud-layer and cluster-layer services have at least 4-8 cores, and the Node Controller hosts often have even more.

The Node Controller hosts run the virtual machines in your cloud. As a result, they benefit from more cores because it allows them to run more virtual machines. How many CPU cores a Node Controller will require depends on:

- How many simultaneous virtual machines you run
- How many virtual CPUs you assign to those virtual machines

By default, each virtual machine CPU core is mapped to a physical core on the Node Controller. The more physical cores that a Node Controller has, the more virtual machines it will support. CPU over commitment is supported and is discussed in another section of this lesson. A Node Controller should have at least 4-8 or more to be useful.

Memory Requirements

Each host requires a minimum of 4GB of memory. However, most hosts will perform better with more memory. Many hosts running cloud-layer and cluster-layer services have at least 8GB of memory, and the Node Controller hosts often have even more.

The Node Controller hosts run the virtual machines in your cloud. As a result, they benefit from more memory because it allows them to run more virtual machines. How much memory a Node Controller will require depends on:

- How many simultaneous virtual machines you run
- How much memory you assign to those virtual machines

To calculate your total memory requirements on a Node Controller, add the memory requirements of the operating system together with the memory requirements of each simultaneously powered-on virtual machine that you expect to run. Memory over commitment is supported and is discussed in another section of this lesson. A Node Controller should have at least 8GB or more to be useful.

Storage Requirements

Eucalyptus hosts require a minimum of 30 GB of storage that is supported by the host operating system. Eucalyptus recommends at least at least 250GB or more on the Walrus, Storage Controllers, and Node Controllers in order to create useful clouds. Larger storage sizes support greater numbers of virtual machines, images, volumes, and snapshots.

Eucalyptus IaaS Subscription offers optional support for certain SAN arrays. These include:

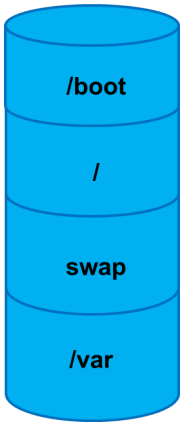
- Dell EqualLogic PS4000 and PS6000 series arrays
- NetApp FAS2000 and FAS6000 series arrays
- EMC VNX series arrays

These arrays can be used by the Storage Controller to provide Eucalyptus Block Store (EBS) volumes to instances running on the Node Controllers. Check the Eucalyptus Web site or contact a Eucalyptus representative to find the latest list of supported arrays.

Please use the latest Eucalyptus documentation to ensure that you have the correct firmware and array management software for your array solution. For example, the EMC VNX series arrays require minimum FLARE and Navisphere Software versions.

Disk and Software Installation

When installing the operating system and configuring the local disk partitions, consider the following recommended minimum sizes.



Configure `/boot` with 250MB.

Configure / with at least 20GB. This is more than enough room for the operating system and provides a fair amount of free space for the `/tmp` directory.

Configure the swap partition to be 1.5 times the size of physical memory.

Configure `/var` with at minimum of 2GB for the operating system, plus extra space for the Eucalyptus software. The `/var` directory is heavily used by the Eucalyptus hosts for such things as the cloud database, log files, Walrus storage, EBS volume creation, image caching, and instance caching. The `/var` directory is used in the following ways by the following hosts.

Cloud Controller	Cloud Database	`/var/lib/eucalyptus/db`	20GB
Cloud Controller	Cloud Controller logs	`/var/log/eucalyptus`	2GB
Walrus	Walrus logs	`/var/log/eucalyptus`	2GB
Walrus	Bucket object storage	`/var/lib/eucalyptus/bukkits`	250-500GB or larger
Storage Controller	Volume storage	`/var/lib/eucalyptus/volumes`	250-500GB or larger (unless SAN)
Cluster Controller	Cluster Controller logs	`/var/log/eucalyptus`	2GB
Node Controller	Image cache	`/var/lib/eucalyptus/instances`	250 GB or larger
Node Controller	Instance cache	`/var/lib/eucalyptus/work`	250GB or larger
Node Controller	Node Controller logs	`/var/log/eucalyptus`	2GB

Network Requirements

Eucalyptus components require a minimum of one, 1GBps Network Interface Card (NIC). In practice, for network isolation and scalability more NICs are recommended. If you are running in either MANAGED or MANAGED-NOVLAN network mode, the Cluster Controller is required to have two NICs - one connected to the Cloud Controller, one connected to the Node Controllers.

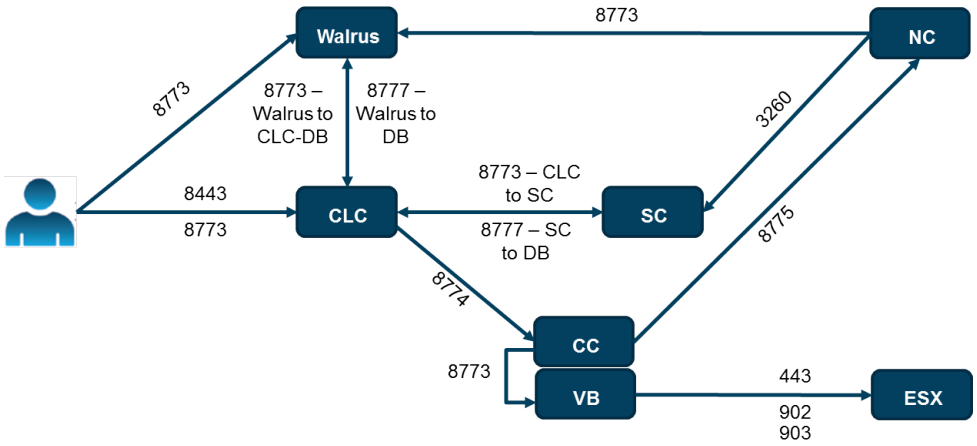
10Gbps NICs might be useful on Cluster Controller hosts running in MANAGED or MANAGED-NOVLAN network mode. In these network modes the Cluster Controllers act as IP routers for the virtual machines running in the cloud, and routing might benefit from the increased bandwidth provided by these NICs.

In a Eucalyptus HA configuration, each functional NIC should be bonded to a second NIC, and each NIC should be connected to a separate physical network.

UDP multicast capability is required for address 228.7.7.3 when Eucalyptus Java-based components are run on separate physical hosts. The Java-based components include the Cloud Controller, Walrus, Storage Controllers, and the VMware Brokers.

Firewall Requirements

The following diagram illustrates the network ports that must be open between Eucalyptus components.



For a proof-of-concept installation, a simple way to open these ports in the Linux firewall would be to disable `iptables`. To disable `iptables` use `system-config-firewall-tui`.

Normally, `iptables` *nat* and *filter* tables are flushed when Eucalyptus starts on the Cluster Controllers. However, it is possible to pre-load a set of firewall rules that will remain in place during Cluster Controller operation. To configure this, add the rules to `iptables` before starting Eucalyptus and then run the command `iptables-save > /var/run/eucalyptus/net/iptables-preload`.

Caution: Performing this operation to define special `iptables` rules that are loaded when Eucalyptus starts could cause Eucalyptus instance networking to fail. Eucalyptus recommends that you only do this if you are completely sure that the pre-loaded rules will not interfere with the operation of Eucalyptus.

Lastly, Eucalyptus is not compatible with SELinux. SELinux should be disabled prior to installing Eucalyptus. The steps to disable SELinux vary with the Linux distribution. For more information see the Installation Guide at http://www.eucalyptus.com/docs/3.1.0/ig.pdf.

Installation Methods

There are several different ways to install Eucalyptus. These include:

- Standard package installation
- Subscription package installation
- FastStart
- Nightly build package installation
- Source file installation

This course focuses on the standard package method.

Eucalyptus can be installed from packages. Package installation is described in the Eucalyptus Installation Guide at http://www.eucalyptus.com/docs. There are two types of package installation. You can install just the standard open source packages or you can install the standard open source packages plus the subscription-only SAN and VMware packages. The two package installation methods are very similar. The main difference is that the subscription-only software can be installed only by using a Eucalyptus-supplied private key and entitlement certificate that enables the installer to access a special software repository that contains the add-on software.

Eucalyptus offers FastStart installation, a suite of installation tools that allows installation and configuration of proof-of -concept or more complex, multi-host Eucalyptus deployments. It is a generalized installer for Eucalyptus that provisions from bare metal to a cloud. You can download a FastStart ISO image from the Eucalyptus Web site or you can build your own FastStart ISO image by downloading the FastStart software from Eucalyptus.

Eucalyptus also allows customers to download and install the latest versions of Eucalyptus which are called the nightly builds. They should be considered unstable, bleeding-edge software and should not be installed in production environments. Nightly builds use the same installation steps as a normal install from packages installation but access a different download path in order to get the latest software.

Eucalyptus can also be installed from source code. Source code is available on Github at *https://github.com/eucalyptus*.

Package Installation

Packages are available for all Eucalyptus supported Linux distributions. Eucalyptus IaaS can be installed from packages whether you have purchased a Eucalyptus subscription or not.

As part of the Eucalyptus IaaS Subscription you will receive an RPM file in email that contains:

- An entitlement certificate and private key that allow you to access and download the subscription-only software
- The subscription software that includes the SAN drivers and the VMware API software
- A GPG key that is used to verify the software's integrity

Internal Software Repository

By default, Eucalyptus software installation depends on having Internet network access. This is because both the Eucalyptus software, as well as the software that Eucalyptus depends on, must be downloaded from various sites on the Internet. However, sometimes it is not practical or acceptable for the Eucalyptus hosts to have Internet access.

Sometimes it might be necessary to create an internal software repository for the Eucalyptus software rather than use the Internet-based repositories. For example, an organization's network security or change control policies might prohibit using an Internet-based repository for installation. Regarding network security, supporting a default installation by opening the necessary ports in their cloud infrastructure to Internet access might violate company security policy. Regarding change control, the software versions on an Internet-based repository can change. Reinstalling servers using these new software versions might violate company change control policies. In both these cases, maintaining an internal, static software repository can be the solution.

Eucalyptus provides guidance about creating an internal, static software repository. For information about creating an internal repository, see Appendix B of the *Eucalyptus Installation Guide* for 3.2 at *http://www.eucalyptus.com/docs*.

Installation Tasks

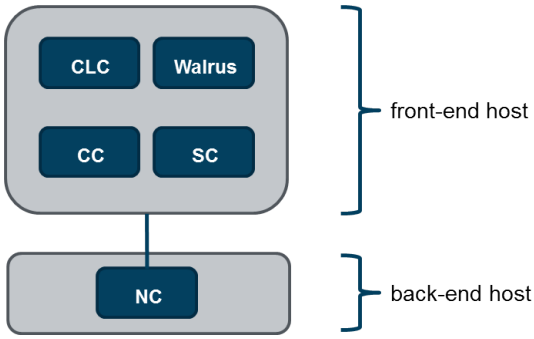
Installation of Eucalyptus IaaS software is a multi-step process. Some of the pre-installation tasks include pre-configuring network bridges on the Node Controllers and installing (or configuring) the Eucalyptus hosts with software repository files that determine where the hosts go to download and install the required software. Installation concludes with the steps that actually install the required software on the Eucalyptus hosts.

Proof-of-Concept Installation

This section illustrates a proof-of-concept installation on CentOS 6.3 running the KVM hypervisor. You will install from packages.

Note: For installation guides on other Linux distributions and versions, refer to the *Installation Guide* at *http://www.eucalyptus.com/docs*.

A basic proof-of-concept installation requires two physical hosts, one for the front-end components (Cloud Controller, Walrus, Cluster Controller, and Storage Controller) and another as a Node Controller.



Ensure that time is synchronized between the Eucalyptus hosts. You will also want to ensure that SELiux is disabled and that the Eucalyptus ports are not blocked by any firewall.

Node Controller Pre-Configuration

SYSTEM, STATIC, and MANAGED-NOVLAN network modes require a KVM bridge to be configured prior to software installation. MANAGED mode, however, will automatically create and manage the necessary network bridges for you.

KVM virtual bridges must be configured manually. The actual configuration steps vary by Linux distribution and version. For more information about configuring a network bridge for your Linux distribution, see the Installation Guide at http://www.eucalyptus.com/docs.

Installing the Release RPM

Installing the release RPM installs the `/etc/yum.repos.d/eucalyptus-release.repo` file. This repo file contains all the information that the `yum` command requires to access and download the core Eucalyptus software.

The following command will download and install the correct repo file for a CentOS 6.3 host. For more information about installing the core Eucalyptus software on other Linux distributions and versions, see the *Installation Guide* for 3.2 at *http://www.eucalyptus.com/docs*.

```
yum install http://downloads.eucalyptus.com/software/eucalyptus/ \
3.2/centos/6/x86_64/eucalyptus-release-3.2.noarch.rpm
```

Installing Community Packages Repo Files

Installing the Enterprise Linux Repository (ELRepo) RPM installs the `/etc/yum.repos.d/elrepo.repo` file on the Walrus. This repo file configures the operating system to download the necessary software to the Walrus from network-based repositories. Eucalyptus uses ELRepo on the Walrus host because ELRepo is used to install Distributed Replicated Block Device (DRBD) software, which is required for Walrus HA.

The following command will download and install the correct elrepo file for a CentOS 6.3 host. For more information about installing the elrepo software on other Linux distributions and versions, see the *Installation Guide* for 3.2 at *http://www.eucalyptus.com/docs*.

```
yum install http://downloads.eucalyptus.com/software/eucalyptus/3.2/ \
centos/6/x86_64/elrepo-release-6.el6.elrepo.noarch.rpm
```

On all hosts, installing the Enterprise Linux (EPEL) RPM installs the `/etc/yum.repos.d/epel.repo` file. This repo file configures the operating system for a number of utilities that Eucalyptus does not package, such as vtun, python26, and various perl libraries.

The following command will download and install the correct EPEL repo file for a CentOS 6.3 host. For more information about installing the EPEL software on other Linux distributions and versions, see the Installation Guide for 3.2 at http://www.eucalyptus.com/docs.

```
yum install http://downloads.eucalyptus.com/software/eucalyptus/3.2/ \
centos/6/x86_64/epel-release-6.noarch.rpm
```

Installing the Euca2ools Repo File

You should install the euca2ools repo file on all hosts that will run either Eucalyptus or euca2ools. This repo file provides all the information that the `yum` command needs to locate and download the euca2ools software. Installing this RPM installs the `/etc/yum.repos.d/euca2ools.repo` file.

The following command will download and install the correct euca2ools repo file for a CentOS 6.3 host. For more information about installing euca2ools software on other Linux distributions and versions, see the *Installation Guide* for 3.2 at *http://www.eucalyptus.com/docs*.

```
yum install http://downloads.eucalyptus.com/software/euca2ools/2.1/ \
centos/6/x86_64/euca2ools-release-2.1.noarch.rpm
```

Subscription Customers

If you are a subscription customer you will receive an RPM package to install on all Eucalyptus hosts. The RPM will install an entitlement certificate, private key, a GPG key, and a repo file with information about where to locate and download the Eucalyptus subscription add-on software.

The entitlement certificate and private key allow a user to access and download the Eucalyptus subscription software. The GPG key verifies the downloaded software integrity.

The entitlement certificate file is named for the license holder and is appended with x.y.z flags that indicates the number of times you have received a certificate and the Eucalyptus software version number. For example, <cert_name>-1.3.2.crt indicates that the file is your first certificate for 3.2. The private key file is named after the license holder but does not include version or numbering information.

The following command will download and install the correct subscription repo file for a CentOS 6.3 host. For more information about installing the subscription-only software on other Linux distributions and versions, see the *Installation Guide* at *http://www.eucalyptus.com/docs*.

```
yum install eucalyptus-enterprise-release-3.2*.noarch.rpm
```

Front-End-Install

Assuming a single host will run all of your front-end components, install Eucalyptus front-end packages using the following commands. The first command installs the Cloud Controller while the second command installs the Cluster Controller, Storage Controller, and the Walrus:

```
yum groupinstall eucalyptus-cloud-controller
yum install eucalyptus-cc eucalyptus-sc eucalyptus-walrus
```

This assumes you have Internet access to the packages. This will also install euca2ools on the front-end server.

If Internet access is not available, you would need to create a local repository with all the necessary packages.

You might install the software on separate physical hosts if architecture is something other than a proof-of-concept architecture used in the lab environment. To install just the Cloud Controller service, run `yum install eucalyptus-cloud-controller`. To install just the Walrus service, run `yum install eucalyptus-walrus`. To install just the Cluster Controller service, run `yum install eucalyptus-cc`. To install just the Storage Controller service, run `yum install eucalyptus-sc`. You install the optional VMware Broker on the same host as the Cluster Controller using `yum install eucalyptus-enterprise-vmware-broker`.

Node Controller Install

Install the Eucalyptus Node Controller software on the back-end host using the following command:

```
yum install eucalyptus-nc
```

This assumes Internet access to the packages. Euca2ools will also be installed as part of the process.

If Internet access is not available, you will need to create a local repository with all the necessary packages.

Installing Only Euca2ools

You might want to install euca2ools on a non-Eucalyptus host. For example, users might need euca2ools running on their Linux laptops so that they can manage their cloud resources. The following commands will install euca2ools, and the supporting EPEL software, on a Linux 6 host. For more information about installing the euca2ools software on other Linux distributions and versions, see the *Installation Guide* at *http://www.eucalyptus.com/docs*.

```
yum install http://downloads.eucalyptus.com/software/eucalyptus/3.2/ \
centos/6/x86_64/epel-release-6.noarch.rpm
```

```
yum install http://downloads.eucalyptus.com/software/euca2ools/2.1/ \
centos/6/x86_64/euca2ools-release-2.1.noarch.rpm
```

```
yum install euca2ools
```

The command above assumes that you have Internet access to the packages.

Lab - Install Eucalyptus 3.2

In this lab exercise you will install a Eucalyptus 3.2 cloud with a single front-end host (Cloud Controller, Walrus, Cluster Controller, and Storage Controller) and a single Node Controller host. Both hosts are preinstalled with CentOS 6.3. The lab exercise assumes that the two hosts have internet access.

The front-end host has two network interface cards: one to connect to the public side of the cloud (for users and management) and one to connect to the private side (to the Node Controller). The public network prefix is 172.16.nn.0/24 and is accessed through the em2 interface. The third octet of the network address varies according to

set of hosts that you are assigned. The private network prefix is 192.168.105.0/24 and is accessed through the em1 interface.

The Node Controller host also has network interfaces on the public and private networks. The Node Controller host's em2 interface is on the public network while the em1 interface is on private network (connected to the front-end host).

Instances on the Node Controller will run in their own private subnetwork. The subnetwork address prefix is 10.110.nn.0/24. The third octet of the subnetwork address will vary according to the set of hosts that you are assigned to.

Please refer to the student IP address handout to get specific IP address information for your set of hosts.

Lab Objectives:

- Prepare the operating systems
- Install the software

Prepare the Operating Systems

In this section of the lab you will prepare the operating systems of the front-end and Node Controller hosts for the installation of the Eucalyptus software.

Note: To ensure proper communication with the Node Controller, the firewall and SELinux on both of your hosts were disabled at installation. However, in a normal installation, you would have to either manually disable these or manually open the Eucalyptus-required ports in the firewall.

Note: On the Node Controller host, a network bridge device with a static IP address was configured for you during installation. The name is *br0*. However, in a normal installation, you would have to configure the bridge yourself.

1. Desktop From your Debian desktop, use Secure Shell (SSH) to log in to the front-end host and the Node Controller host. (If necessary, ask your instructor how to access the Debian desktop and open xterm windows in order to use SSH. Once in an xterm window, you can increase the font size by simultaneously pressing the Control key and the right mouse button, and selecting a larger font size from the drop-down menu.)

```
# ssh <front_end_public_IP>            -in one xterm window
# ssh <node_controller_public_IP>      -in another xterm window
```

2. Front End On the front-end host, change the root password in order to prevent accidental log in by other students. Change the password to `passwordNN`, where *NN* is the number of your student pod. For example, if your instructor assigned you to Pod01, set your root password to password01.

```
# passwd
```

3. Node On the Node Controller host, change the root password in order to prevent accidental log in by other students. Change the password to `passwordNN`, where *NN* is the number of your student pod. For example, if your instructor assigned you to Pod01, set your root password to password01.

```
# passwd
```

4. Front End Synchronize the time and date of the front-end host with an NTP server.

```
# ntpdate pool.ntp.org
# service ntpd start
# chkconfig ntpd on
# ps ax | grep ntp
```

```
# hwclock --systohc
```

5. Node Synchronize the time and date of the Node Controller host with an NTP server.

```
# ntpdate pool.ntp.org
# service ntpd start
# chkconfig ntpd on
# ps ax | grep ntp
# hwclock --systohc
```

Install the Software

Eucalyptus provides software packages or source files for installation. You will install Eucalyptus from software packages. Installation packages are available for different operating system environments and CPU architectures. In this lab exercise you will use the relevant 3.2 installation packages for Centos 6.3 on an x86_64 CPU architecture.

1. Front End On your front-end host install the release RPM file. Verify that the `/etc/yum.repos.d/eucalyptus-release.repo` file was installed.

```
# yum install http://downloads.eucalyptus.com/software/eucalyptus/3.2/ \
centos/6/x86_64/eucalyptus-release-3.2.noarch.rpm
# ls /etc/yum.repos.d
```

2. Node On your Node Controller host install the release RPM file. Verify that the `/etc/yum.repos.d/eucalyptus-release.repo` file was installed.

```
# yum install http://downloads.eucalyptus.com/software/eucalyptus/3.2/ \
centos/6/x86_64/eucalyptus-release-3.2.noarch.rpm
# ls /etc/yum.repos.d
```

3. Front End On your front-end host install the euca2ools RPM file. Verify that the `/etc/yum.repos.d/euca2ools.repo` file was installed.

```
# yum install \
http://downloads.eucalyptus.com/software/euca2ools/2.1/centos/6/x86_64/ \
euca2ools-release-2.1.noarch.rpm
# ls /etc/yum.repos.d
```

4. Node On your Node Controller host install the euca2ools RPM file. Verify that the `/etc/yum.repos.d/euca2ools.repo` file was installed.

```
# yum install \
http://downloads.eucalyptus.com/software/euca2ools/2.1/centos/6/x86_64/ \
euca2ools-release-2.1.noarch.rpm
# ls /etc/yum.repos.d
```

5. Front End On your front-end host install the ELREPO RPM file that is required on the machine hosting the Walrus service. Verify that the `/etc/yum.repos.d/elrepo.repo` file was installed.

```
# yum install http://downloads.eucalyptus.com/software/eucalyptus/3.2/ \
centos/6/x86_64/elrepo-release-6.noarch.rpm
# ls /etc/yum.repos.d
```

6. Front End On your front-end host install the EPEL RPM file. Verify that the `/etc/yum.repos.d/epel.repo` file was installed.

```
# yum install http://downloads.eucalyptus.com/software/eucalyptus/3.2/ \
centos/6/x86_64/epel-release-6.noarch.rpm
# ls /etc/yum.repos.d
```

7. Node On your Node Controller host install the EPEL RPM file. Verify that the `/etc/yum.repos.d/epel.repo` file was installed.

```
# yum install http://downloads.eucalyptus.com/software/eucalyptus/3.2/ \
centos/6/x86_64/epel-release-6.noarch.rpm
# ls /etc/yum.repos.d
```

8. Front End On the front-end host, install the Eucalyptus Cloud Controller software. Euca2ools will also be installed.

```
# yum groupinstall eucalyptus-cloud-controller
```

9. Front End On the front-end host, install the remaining Eucalyptus software services.

```
# yum install eucalyptus-cc eucalyptus-sc eucalyptus-walrus
```

10. Node On the Node Controller host, install the Eucalyptus Node Controller software. Euca2ools will also be installed.

```
# yum install eucalyptus-nc
```

Post-Installation Tasks

Installing the software is only the first step in configuring an operational cloud. After installation there are a number of post-installation configuration tasks. The number and types of tasks will vary for each installation. Post-installation tasks are determined by a number of factors including:

- The number of Eucalyptus software components
- Architecture of the Eucalyptus cloud
- The operating system running of the Eucalyptus hosts
- The hypervisors chosen
- Whether or not there is an optional SAN
- The network mode

This course does not install all possible options. For more information about options not covered in this course, see the *Installation Guide* at *http://www.eucalyptus.com/docs*.

Loop Devices

Both the Storage Controller and the Node Controller hosts use Linux loop devices. The Node Controller uses them to build instances. The Storage Controller uses them to provide volume access. If the Node Controller runs out of loop devices then instances will fail to launch. If the Storage Controller runs out of loop devices then it will fail to create EBS volumes.

Eucalyptus recommends a minimum configuration of 50 loop devices. If you have fewer than 50, a startup script will provide a warning message.

For Centos and RHEL 6, 256 loop devices are automatically created when the Eucalyptus components are started the first time. For more information about changing the number of loop devices in the different Linux distributions, see the *Installation Guide* for 3.2 at *http://www.eucalyptus.com/docs*.

Configure the Network Mode

To configure your network mode, edit the appropriate `VNET_*` entries in the `/etc/eucalyptus/eucalyptus.conf` file on the Cluster Controller and Node Controller hosts.

Note: For more information, see the *Installation Guide* for 3.2 at *http://www.eucalyptus.com/docs*.

Any network change to a running Cluster Controller requires that it be reset by performing a clean restart. This clears any existing network configurations you might have regardless of whether or not they are in use. Eucalyptus recommends that you only perform this type of restart when all instances have been terminated. To perform a clean restart, run the following command:

```
service eucalyptus-cc cleanrestart
```

Eucalyptus DNS Names

The Cloud Controller can be configured as a DNS nameserver for instances and Walrus buckets. The DNS host name for the Cloud Controller is eucalyptus.<eucadomain>.<parentdomain>. The DNS host name for the Walrus is walrus.<eucadomain>.<parentdomain>. The <eucadomain> is the domain for which the Cloud Controller is authoritative. The <parentdomain> is the domain for which the organization's DNS nameserver is authoritative.

Instance are assigned DNS host names according to the following pattern:

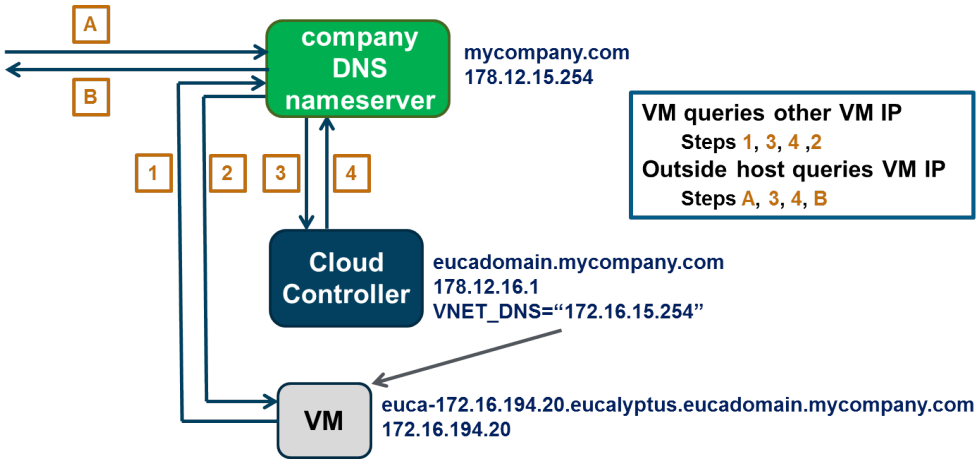
euca-A.B.C.D.eucalyptus.<eucadomain>.<parentdomain> where A.B.C.D is the IP address of the instance.

Walrus buckets are assigned DNS host names according to the following pattern:

<bucketname>.walrus.<eucadomain>.<parentdomain>

As an example, assume that the parent domain is mycompany.com. Assume the eucadomain was configured with the name cloud. The public IP address range for the instances is 172.16.194.20 through 172.16.194.40. In this configuration, an instance might be assigned the DNS domain name euca-172.16.194.20.eucalyptus.cloud.mycompany.com. A bucket named centos would be assigned the DNS domain name centos.walrus.cloud.mycompany.com.

Eucalyptus DNS Queries



The company DNS nameserver must be configured to forward DNS requests to the Cloud Controller because the Cloud Controller is authoritative for the instance and bucket names in the eucalyptus domain. The VNET_DNS="" entry in the `/etc/eucalyptus/eucalyptus.comf` file is configured to point to the company DNS nameserver for two reasons. First, in many cloud environments the instances do not have access to the Cloud Controller and therefore cannot use it as a DNS nameserver. Second, many of the DNS queries will not be for the hostnames of other instances, but for hostnames on the public network. The company DNS nameserver is in a better position to resolve these hostname requests.

If you are supporting Windows instances that are using Active Directory authentication, the nameserver must be able to resolve the DNS names of the AD domain controllers.

VNET_DNS is not available in SYSTEM network mode. In this case you will need to configure the instances to use the DNS nameserver supplied by the corporate DHCP server.

Configuring DNS

Use the following procedure to enable DNS on your Cloud Controller.

1. `euca-modify-property –p system.dns.dnsdomain=<eucadomain>.<parentdomain>`
2. `euca-modify-property -p bootstrap.webservices.use_instance_dns=true`
3. Enter the IP address of the company <parentdomain> nameserver as VNET_DNS="parent_domain_nameserver_IP" in `/etc/eucalyptus/eucalyptus.conf`.
4. Configure the parent domain nameserver to point to the Cloud Controller and the <eucadomain>.

The first `euca-modify-property` command configures the Cloud Controller with the domain name for which it is authoritative. The <parentdomain> is the domain name for which the company DNS nameserver is authoritative. The second `euca-modify-property` command enables the DNS service on the Cloud Controller. The VNET_DNS entry in the `eucalyptus.conf` file configures the Cloud Controller with the IP address of its parent domain name server.

Note: There is also a cloud property named *system.dns.nameserver* but it does not need to be changed in order for DNS resolution to work.

Note: The DNS nameserver must be able to attach to port 53. If dnsmasq is configured to start at boot, it will attach to port 53 and interfere with the Eucalyptus nameserver. Disable dnsmasq if necessary.

There are DNS configuration file examples in the Installation Guide for 3.2 at http://www.eucalyptus.com/docs.

Optional Configuration

There are several optional configuration steps, which depend on the features in use in your Eucalyptus environment. These optional steps include:

- CPU/memory over commitment
- Configuring Virtio paravirtualized network and disk devices for KVM
- Configuring VMware support
- Configuring SAN support
- Configuring HA Eucalyptus

This course does not install all possible options. For more information about options not covered in this course, see the *Installation Guide* for 3.2 at *http://www.eucalyptus.com/docs*.

CPU Over Commitment

By default, each physical CPU core on a Node Controller is available for assignment to an instance. For example, a single-CPU instance would be assigned a single physical CPU core whereas a quad-CPU instance would be assigned four physical CPU cores. If a Node Controller has eight-cores it can support eight single-CPU instances, or four dual-CPU instances, or two quad-CPU instances. If the physical CPU cores are hyperthreaded, it doubles the number of physical cores available for assignment to instances. If you have four Node Controllers, each with eight physical hyperthreaded cores, then your cloud could run a maximum of 64 single-CPU instances.

However, it is possible to configure each individual Node Controller to behave as though they have more physical CPU cores available they actually do. On each Node Controller modify the `MAX_CORES="n"` entry in the `/etc/eucalyptus/eucalyptus.conf` file if you wish to configure the cloud to support CPU over commitment. Set the value `n` to be equal to the number of physical CPU cores you want the Node Controller to assume that it has.

If a Node Controller's CPUs are over committed, Eucalyptus relies on the underlying hypervisor for fair CPU resource scheduling.

.

Memory Over Commitment

By default, all available memory on a Node Controller is available for provisioning to instances. Optionally, memory resources can be over committed per Node Controller. Memory over commitment is controlled by the `MAX_MEM="n"` entry each Node Controller's `/etc/eucalyptus/eucalyptus.conf` file. Set the value `n` to be equal to the number of megabytes of memory you want the Node Controller to assume that it has. If a Node Controller's memory is over committed, Eucalyptus relies on the underlying hypervisor for fair memory resource scheduling. Be aware that if instances are actively using more memory than is physically available, then paging from disk will occur and application performance will likely be reduced. The more paging that occurs, the more that performance will typically be degraded.

Note: Eucalyptus does not support storage over commitment.

Start the Cloud Controller, Walrus, and Storage Controller

After the Cloud Controller software has been installed but before you start the Cloud Controller service the first time, you need to initialize the Cloud Controller database. Initializing the database only take a minute or so to complete. The database stores configuration, runtime, and resource usage information. Manual maintenance of the database is not normally necessary.

To launch the cloud, initialize the database on the Cloud Controller with the following command:

```
euca_conf --initialize
```

Once the database is initialized, start the Cloud Controller service by running the following command on the Cloud Controller:

```
service eucalyptus-cloud start
```

This command also starts the Walrus, Storage Controller, and VMware Broker if they are installed on the same host.

Note: As an alternative, the following command may be used:

```
/etc/init.d/eucalyptus-cloud start
```

Start the Cluster Controller and Node Controller

Once the Cloud Controller and other front-end components have been started, you can start the Cluster Controller by running the following command on the Cluster Controller:

```
service eucalyptus-cc start
```

Note: An alternative command:

```
/etc/init.d/eucalyptus-cc start
```

Note: In our example installation, the Cluster Controller is located on the front-end host with the Cloud Controller, Walrus, and Storage Controller. If the Walrus and Storage Controller were on different hosts, they would need to be started prior to starting the Cluster Controller.

Once the Cluster Controller is running, start your Node Controllers by running the following command on the Node Controllers:

```
service eucalyptus-nc start
```

Note: An alternative command:

```
/etc/init.d/eucalyptus-nc start
```

A quick way to determine if the Eucalyptus services are running is to run `netstat` on the various hosts and look to see whether ports have been allocated to the services. Expected outcomes include:

- The Cloud Controller is listening on ports 8443 and 8773
- Walrus is listening on port 8773
- The Storage Controller is listening on port 8773
- If you are using the subscription only VMware Broker, it is listening on port 8773
- The Cluster Controller is listening on port 8774
- The Node Controller is listening on port 8775
- Log files are being written to `/var/log/eucalyptus/`

Starting and Stopping Cloud Services

Once the Eucalyptus services have been started and registered, it is important to start and stop the cloud services in the proper order. This ensures proper operation by maintaining the proper state information between the services. The following table lists the proper start up and shut down orders:

1. Start the CLC	1. Stop instances
2. Start the Walrus	2. Stop the NC
3. Start the SC	3. Stop the CC
4. Start the VB	4. Stop the VB
5. Start the CC	5. Stop the SC
6. Start the NC	6. Stop the Walrus
7. Start instances	7. Stop the CLC

Register Components

Eucalyptus services must be registered as part of the post-installation tasks. You must register the Walrus, Cluster Controller, and Storage Controller, with the Cloud Controller. The Node Controllers must be registered with their Cluster Controller. The VMware Broker must also be registered with its Cluster Controller, however, a VMware Broker is not used in the training environment.

Registration allow these services to not only become aware of each other, but to exchange information that allows secure communication.

Register a Walrus

To register the Walrus, enter the following command on the Cloud Controller:

```
euca_conf --register-walrus --partition walrus \
```

```
--host <public_Walrus_IP_addr> --component <unique_name>
```

Note: For example:

```
euca_conf --register-walrus --partition walrus --host 173.16.45.12 \
--component walrus-hostA
```

The following describes the command options:

- `--partition`: The partition the component will belong to. The partition is analogous to an availability zone or cluster. Walrus is not part of any partition so the placeholder text *walrus* is used instead.
- `--component`: The name ascribed to the Walrus. This is the name used to identify the Walrus in a human friendly way. This name is used when reporting system state changes which require attention. It must be globally-unique with respect to other component registrations.
- `--host`: The public IP address of the Walrus host.

Note: The component name *walrus-hostA* was chosen to accommodate Eucalyptus HA where the redundant Walrus service might be registered as *walrus-hostB*. Notice that each component name is unique in the cloud and follows the form of *service_name-host_location*.

Register a Cluster Controller

To register the Cluster Controller enter the following command on the Cloud Controller:

```
euca_conf --register-cluster --partition <partition_name> \
   --host <cluster_controller_public_IP_addr> --component <unique_name>
```

Note: For example:

```
euca_conf  --register-cluster  --partition  clusterA   \
   --host  173.16.45.14 --component  cc-hostC
```

The following describes the command options:

- `--partition`: The name the administrator assigns to the cluster. You should choose a name that describes your cluster. For example, the name could describe the cluster's location, purpose, or performance characteristics.
- `--component`: The name ascribed to the Cluster Controller. This is the name used to identify the Cluster Controller in a human friendly way. This name is used when reporting system state changes which require attention. It must be globally-unique with respect to other component registrations.
- `--host`: The public IP address of the Cluster Controller.

Note: The component name *cc-hostC* was chosen to accommodate Eucalyptus HA where the redundant Cluster Controller service might be registered as *cc-hostD*. Notice that each component name is unique in the cloud and follows the form of *service_name-host_location*.

Register a Storage Controller

To register the Storage Controller enter the following command on the Cloud Controller:

```
euca_conf --register-sc --partition <partition_name> \
   --host <storage_controller_public_IP_addr> --component <unique_name>
```

Note: For example:

```
euca_conf  --register-sc  --partition  clusterA   \
   --host  173.16.45.12 --component  sc-hostE
```

The following describes the command options:

- `--partition`: The name the administrator assigns to the cluster. This should match the accompanying Cluster Controller registration partition label.

- `--component`: The name ascribed to the Storage Controller. This is the name used to identify the Storage Controller in a human friendly way. This name is used when reporting system state changes which require attention. It must be globally-unique with respect to other component registrations.
- `--host`: The public IP address of the Storage Controller.

Note: The component name *sc-hostE* was chosen to accommodate Eucalyptus HA where the redundant Storage Controller service might be registered as *sc-hostF*. Notice that each component name is unique in the cloud and follows the form of *service_name-host_location*.

Register Node Controllers

To register a Node Controller enter the following command on the Cluster Controller (*not* the Cloud Controller, unless the two services are installed on the same physical host):

```
euca_conf –-register-nodes=<node_controller_IP_addr>
```

Note: For example:

```
euca_conf --register-nodes=174.12.6.6
```

The IP address should be the IP address of the Ethernet interface that connects to the Node Controller.

Node Controllers belong to a specific partition, which is controlled by a specific Cluster Controller. Because the Node Controllers are registered with their Cluster Controller, there is no need to include a `--partition` option in this registration command.

Node Controllers are also not redundant in a Eucalyptus HA environment. Therefore, there is no need for a `--component` option in this registration command.

Because there are potentially many Node Controllers in each cluster, there is no `--host` option in this registration command. Instead, to simplify the registration process the administrator can register all Node Controllers at once by providing a space-separated list of IP addresses enclosed in double quotes.

Backup the Cloud Configuration

Once a cloud has been configured, you can back up the configuration so that you can recover the cloud in case of hardware problems or human error. You need to back up all user data and customizations that are not part of a standard installation. You can use standard Linux utilities to back up the files and directories # like `cp` or `tar` # just be sure to capture the original file and directory ownerships and permissions.

You need to back up the following:

Host	Directory	Contents
All hosts	`/etc/eucalyptus/eucalyptus.conf`	Main configuration file
Cloud Controller	`/var/lib/eucalyptus/db`	Cloud database
All hosts	`/var/lib/eucalyptus/keys`	Keys to authenticate cloud services
Walrus	`/var/lib/eucalyptus/bukkits`	Bucket storage - could be located on NFS or SAN array
Storage Controller	`/var/lib/eucalyptus/volumes`	EBS volumes - could be located on SAN array

Recover the cloud by reinstalling the software from packages and then restore the backed-up files and directories to their original locations.

Download Admin Credentials

Command line (CLI) and graphic user interface (GUI) access to the cloud is authenticated by credentials. Even an administrator must provide credentials before running CLI commands or using a GUI to manage the cloud. To generate and download administrator credentials, run the following commands:

```
# cd /root
# mkdir .euca
# cd .euca
# euca_conf –-get-credentials admin.zip
# unzip admin.zip
# chmod 700 .euca
# chmod 600 .euca/*
# source ./euarc
```

The files in `admin.zip` are

- `euca2-admin-hhhhhhhh-cert.pem`,
- `eucarc`,
- `jssecacerts`,
- `cloud-cert.pem`,
- `euca2-admin-hhhhhhhh-pk.pem`, and
- `iamrc`.

Note: Where *hhhhhhhh* is eight hexadecimal digits.

Two types of credentials are issued by EC2- and S3-compatible services (such as Eucalyptus):

- x.509 PEM-encoded certificates
- private keys

Changing permissions on the `.euca` directory and its files protects them against unauthorized access.

Note:

Credentials can be downloaded for normal cloud users too. As root on the Cloud Controller, use the command syntax `/usr/sbin/euca_conf --cred-account <account> --cred-user <user_name> --get-credentials <filename>.zip`.

Euca2ools Operation

Euca2ools requires encrypted credentials (keys, or certificates and keys) to authenticate user identity. Euca2ools also requires the URLs of the Cloud Controller and the Walrus unless these services reside on the local host. All this information can be supplied using command-line arguments every time a euca2ools command is issued. However, this would be a time-consuming and error-prone way to use euca2ools.

The `eucarc` file simplifies command-line operations by configuring environment variables for keys, certificates, and URLs. Euca2ools command have been written in such a way that they will check for these environment variables and use them if they exist. This removes the requirement of providing this information using command-line arguments.

If the `eucarc` file has not been read by the user's shell, you will get an error message when running euca2ools commands. To fix this in the current user shell, force the shell to read the file using the following command:

```
# source /root/.euca/eucarc
```

To ensure that this file is always read at shell startup, edit the shell startup file, the `/root/.bashrc` file for example, and add the following entry:

```
source /root/.euca/eucarc
```

Configure Storage Controller Storage

A Storage Controller can use several different types of storage to build and manage EBS volumes. There are two broad Storage Controller configurations: file system-backed and SAN-backed. File system-backed can be further dividing into those using files and loopback devices, and those based on a LVM.

Before configuring any volumes, you must configure the Storage Controller with the storage manager service that matches the type of storage that will be used for volumes. However, you cannot configure the Storage Controller storage manager until you have cloud administrator credentials.

The command to configure the storage manager is:

```
# euca-modify-property -p
 <cluster_name>.storage.blockstoragemanager=<manager_name>
```

The possible types of storage managers are:

- `das` # local disk managed by an LVM, manipulates volumes directly using LVM commands
- `overlay` # local disk with a file system and no LVM, manipulates volumes using file commands and loop devices
- `emc-vnx` # support EMC VNX series SAN arrays
- `emc-vnx-fastsnap` # support EMC VNX series SAN arrays with FLARE 5.32 (August 2012) or above
- `equallogic` # supported Dell Equallogic SAN arrays (PS4000 and PS6000 series arrays)
- `netapp` # support Netapp SAN arrays (FAS2000 and FAS6000 series arrays)

The difference between the two EMC managers is related to when a snapshot of a volume becomes consistent with the volume. With emc-vnx a snapshot of a volume becomes consistent with the volume at the end of the snapshot process. With emc-vnc-fastsnap the snapshot becomes consistent with the volume very early in the snapshot process.

Until you configure the Storage Controller storage manager, the Storage Controller service will report a status of BROKEN in the output of `euca-describe-services` or `euca_conf --list-scs`. Once you configure a storage manager the serivce status will automatically switch to ENABLED.

Verify Cloud Resources

A functioning cloud should provide CPU, memory, storage, and public IP address resources.

Note: A cloud configured in SYSTEM or STATIC network mode would not provide public IP address resources.

To verify that CPU, memory, and storage resources are available for running instances, run the following command:

```
euca-describe-availability-zones verbose
```

```
AVAILABILITYZONE        |- vm types     free / max   cpu   ram  disk
AVAILABILITYZONE        |- m1.small     0004 / 0004   1    128     2
AVAILABILITYZONE        |- c1.medium    0004 / 0004   1    256     5
AVAILABILITYZONE        |- m1.large     0002 / 0002   2    512    10
AVAILABILITYZONE        |- m1.xlarge    0002 / 0002   2   1024    20
AVAILABILITYZONE        |- c1.xlarge    0001 / 0001   4   2048    20
```

Non-zero values in the *free* column indicate that the Node Controller is providing resources to the cloud. If all values are zero, you will not be able to launch instances. The likely root cause is a misconfiguration of the cloud. For example, make sure that the Node Controller is properly registered with the Cluster Controller using the IP address of the interface that is attached to the Cluster Controller.

To verify that public IP addresses are available for running instances, run the following command:

```
euca-describe-addresses
```

The result should be a list of public IP addresses.

```
ADDRESS 172.16.164.20   nobody  standard
ADDRESS 172.16.164.21   nobody  standard
. . .
```

If you do not see any public IP address then it is likely that the cloud is misconfigured. For example, check the network configuration parameters (VNET_*) in the `/etc/eucalyptus/eucalyptus.conf` file on Cluster Controller and Node Controller hosts. If you modify the network configuration remember to run the command `service eucalyptus-cc cleanrestart`.

Lab - Post-Installation Tasks

In this lab exercise you will perform various post-installation configuration tasks. You will also download cloud administrator credentials and verify that your cloud has resources available to create instances.

Lab Objectives:

- Configure the network mode
- Start Eucalyptus services
- Register Eucalyptus services
- Download cloud administrator credentials
- Configure a storage manager
- Verify cloud resources exist

Configure the Network Mode

In this section of the lab you will configure Eucalyptus to run in the MANAGED-NOVLAN network mode.

1. Front End On the front-end host, configure the network mode of your cloud by editing the appropriate VNET parameters in the `/etc/eucalyptus/eucalyptus.conf file`. You can use either the `vi` or `nano –w` editor.

Note: Be sure to uncomment (remove the leading # character) the VNET parameters where appropriate.

```
# vi /etc/eucalyptus/eucalyptus.conf

VNET_MODE="MANAGED-NOVLAN"
VNET_PRIVINTERFACE="em1"
VNET_PUBINTERFACE="em2"
VNET_BRIDGE=”br0”
VNET_PUBLICIPS="<Public_IP_addr_range_listed_on_your_handout>"
VNET_SUBNET="<VNET_SUBNET_addr_listed_on_your_handout>"
VNET_NETMASK="255.255.254.0"
VNET_ADDRSPERNET="32"
VNET_DNS="8.8.8.8"
VNET_DHCPDAEMON=”/usr/sbin/dhcpd41”
```

Note: Make sure that all other VNET parameters in the file are commented out (start with a leading # character).

2. Front End On the front-end host, use Secure Copy (`scp`) to copy the `/etc/eucalyptus/eucalyptus.conf` file from the front-end host to the Node Controller host. Use the Node Controller's private IP address as shown on your student handout.

Note: In our lab configuration the same `eucalyptus.conf` file can be used on both the front-end and Node Controller hosts so rather than edit the file twice, you can just copy it from one host to the other host.

```
# scp /etc/eucalyptus/eucalyptus.conf \
<node_controller_private_IP>:/etc/eucalyptus/eucalyptus.conf
```

3. Node On the Node Controller host, view the `/etc/eucalyptus/eucalyptus.conf` file. Verify that the the file was copied correctly to the host.

```
# more /etc/eucalyptus/eucalyptus.conf
```

Start Eucalyptus Services

In this section of the lab you will start the Eucalyptus services on the front-end and Node Controller hosts.

1. Front End On the front-end host, perform a first-time initialization of the Cloud Controller database. This command will take a few minutes to complete.

```
# euca_conf --initialize
```

2. Front End On the front-end host, start the front-end services (Cloud Controller, Walrus, Cluster Controller, Storage Controller).

```
# service eucalyptus-cloud start
# service eucalyptus-cc start
```

3. Front End On the front-end host, verify that the front-end services are running.

```
# service eucalyptus-cloud status
# service eucalyptus-cc status
```

4. Node On the Node Controller host, start the Node Controller service.

```
# service eucalyptus-nc start
```

5. Node On the Node Controller host, verify that the Node Controller service is running.

```
# service eucalyptus-nc status
```

Register Eucalyptus Services

Front End

In this section of the lab you will register the Walrus, Cluster Controller, and Storage Controller services with the Cloud Controller. You will also register the Node Controller service with the Cluster Controller.

1. On the front-end host, register the Walrus service with the Cloud Controller. The Walrus public IP address is the same as the front-end public IP address.

```
# euca_conf –-register-walrus --partition walrus \
–-host <walrus_public_IP> –-component walrus00
```

   Answer `yes` when asked if you want to continue. Enter the root password of the Walrus host (front-end host) when prompted. The root password should be `passwordNN` where *NN* is the number of your student pod.

2. On the front-end host, register the Cluster Controller service with the Cloud Controller. Name your cluster *cluster1*. The Cluster Controller public IP address is the same as the front-end host public IP address.

```
# euca_conf –-register-cluster --partition cluster1 \
–-host <cluster_controller_public_IP> –-component cc00
```

   Enter the root password of the Cluster Controller host (front-end host) when prompted. The root password should be `passwordNN` where *NN* is the number of your student pod.

3. On the front-end host, register the Storage Controller with the Cloud Controller. The Storage Controller public IP address is the same as the front-end public host IP address.

```
# euca_conf –-register-sc --partition cluster1 \
–-host <storage_controller_public_IP> –-component sc00
```

   Enter the root password of the Storage Controller host (front-end host) when prompted. The root password should be `passwordNN` where *NN* is the number of your student pod.

   Note: You will receive a message about the need to "choose a storage back end" when you register the Storage Controller. You will choose the storage back end later in this lab exercise.

4. On the front-end host, register the Node Controller with the Cluster Controller. Use the Node Controller private IP address listed on your student handout.

```
# euca_conf –-register-nodes=<node_controller_private_IP>
```

   Enter the root password of the Node Controller host when prompted. The root password should be `passwordNN` where *NN* is the number of your student pod.

5. On the front-end host, verify that all Eucalyptus services are registered.

```
# euca_conf –-list-walruses
# euca_conf –-list-scs
# euca_conf –-list-clusters
# euca_conf –-list-nodes
```

   Note: The `STORAGECONTROLLER` service will report its status as `BROKEN`. This will change to `ENABLED` once you configure a back-end storage manager later in the lab.

Download Cloud Administrator Credentials

Front End

In order to manage and use your cloud using euca2ools, it is necessary to supply the proper credentials to the Cloud Controller each time you enter a command. This can be done by adding authentication arguments to each command at the command line or by setting authentication environment variables that are automatically used each time a command is run. Setting and using environment variables is simpler. In this section of the lab, you will download and install files that will automatically set the necessary environment variables for the cloud administrator user named *admin*.

1. On the front-end host, generate a set of credentials for the cloud administrator named *admin*.

```
# cd /root
# mkdir .euca
# cd .euca
# euca_conf --get-credentials admin.zip
# ls
```

Note: Any time you download credentials; you are requesting a *new* set of credentials. You are *not* retrieving an existing set of credentials. There is no way to retrieve an existing set of credentials.

2. Unpackage the `admin.zip` file and secure the directory and its contents using the permissions shown below.

```
# unzip admin.zip
# chmod 700 /root/.euca
# chmod 600 /root/.euca/*
```

3. Using either the `vi` or `nano -w` editor, edit root's bash shell configuration file and add the `source` command to the *end* of the file that will force all bash shells to execute the commands listed in the `eucarc` file. Do not close the editor when finished as you will continue to edit this file in the next lab step.

```
# vi /root/.bashrc

source /root/.euca/eucarc
```

4. With the `/root/.bashrc` file still open for editing, add a new alias for the `ssh` command to the end of the list of current `alias` commands. Save your changes and exit the file when finished.

```
alias  ssh='ssh -o UserKnownHostsFile=/dev/null -o \
 StrictHostKeyChecking=no'
```

Note: Adding this alias will cause the `ssh` command to automatically run with options that configure it to *not* update or check the `.ssh/known_hosts` file. Preventing the use of the `known_hosts` file when using `ssh` in lab will be more convenient. When the public key associated with an IP address frequently changes, as will be the case when you create and terminate instances which reuse the same IP address, you would have to remove the IP address/public key entry in the `known_hosts` file before being able to `ssh` to a new instance. Preventing the normal use of the `known_hosts` file will remove the need to manually edit and remove the IP address/public key entry each time an IP address is reused for a new instance.

5. On the front-end host, force root's bash shell to immediate reread the contents of the `/root/.bashrc` file.

```
# source /root/.bashrc
```

6. Verify that the `.bashrc` file was reread by checking for the presence of EC2 environment variables in your shell.

```
# env | grep EC2
```

Configure a Storage Manager

The Storage Controller service can manage the creation of volumes on several different types of back-end storage. In this section of the lab you will configure the Storage Controller to use the correct storage manager software for the type of back-end storage used in the lab environment. The lab environment uses local disks with a file system.

1. On the front-end host, use the `euca-modify-property` command to configure the correct storage manager for the Storage Controller service. The storage manager you will use is `overlay`. The Storage Controller should have been configured in a cluster (partition) named *cluster1*.

```
# euca-modify-property -p cluster1.storage.blockstoragemanager=overlay
```

2. Verify that the Storage Controller service now shows a status of `ENABLED`. You can use either the `euca_conf` command or the `euca-describe-services` command.

```
# euca_conf --list-scs
# euca-describe-services
```

Note: The output of the `euca-describe-services` command will be explained in a later course module.

Verify Cloud Resources Exist

In this section of the lab you will verify that your cloud is aware that CPU, memory, storage, and IP address resources exist for running instances. Verifying that the cloud is aware of these resources provides a quick test of cloud functionality prior to registering images and launching instances. Running the commands to test for these resources also provides a quick test of the functionality of the euca2ools command-line tools.

1. On the front-end host, verify the availability of CPU, memory, and storage resources. Specifically look for *non-zero* numbers in the `free` column in the screen output.

```
# euca-describe-availability-zones verbose
```

Note: The `free` column denotes the maximum number of instances that can be started for each of the five vmtypes listed in the screen output. Notice that as the size of the vmtypes gets larger, fewer instances can be run because the larger vmtypes consume more physical resources.

Note: Seeing non-zero numbers in the `free` column is a good indicator that the front-end and Node Controller services are configured and communicating correctly. If the front-end cannot correctly communicate with the Node Controller, then no CPU, memory, or storage resource will be shown as available.

2. On the front-end host, verify the availability of public IP address resources in the screen output.

```
# euca-describe-addresses
```

Note: Unless a user explicitly requests that an instance not receive a public IP address at startup, each instance will automatically be assigned a public IP address. Therefore, it is important that these addresses are available.

Note: Seeing a list of public IP addresses is a good indicator that the network configuration in the hosts' `eucalyptus.conf` files is correct.

Management Tools

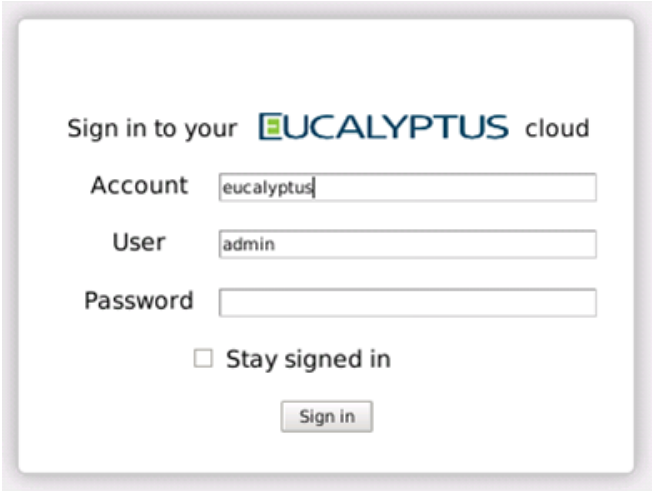
There are many ways to interact with the Eucalyptus cloud. Eucalyptus provides a Web-based Eucalyptus Administrator Console utility to help you manage your environment, as well as a collection of command-line administration tools called administrator tools. Eucalyptus also provides a set of command-line user tools called euca2ools. There is also a Web-based Eucalyptus User Console that allows normal cloud users to manage their cloud resources. Lastly, there are also a large number of third-party tools available to help configure, manage, monitor, and create images in a Eucalyptus environment .

Eucalyptus Administrator Console

The Eucalyptus Administrator Console is a Web-based interface for managing Eucalyptus services, IAM identities, and access control policies, and quotas. It provides and easy-to-use graphical management tool for cloud administrators. The Eucalyptus Administrator Console is accessed through the URL https://<CLC_public_IP>:8443.

Log In to the Eucalyptus Administrator Console

After typing the URL to the Eucalyptus Administrator Console in your Web browser you will see the following log in screen.

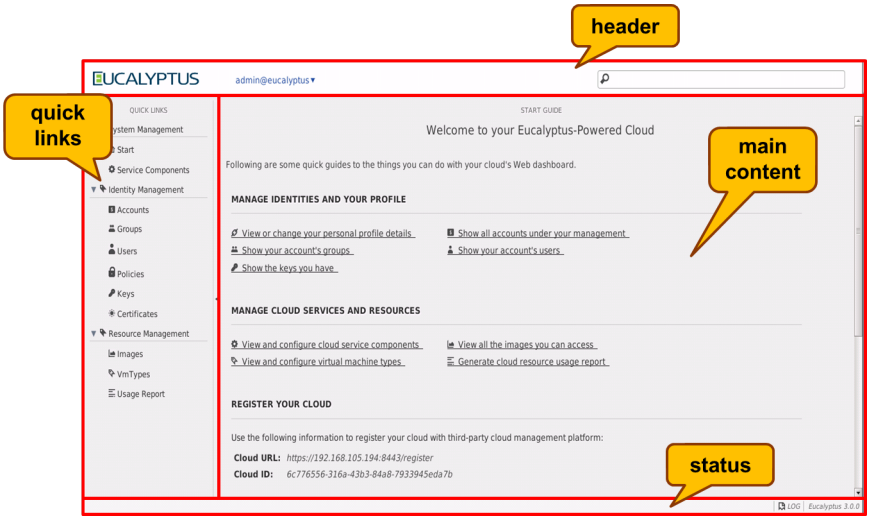


By default, the only account that exists after installation is the *eucalyptus* account. The default administrative user is *admin* and the default password is also *admin*.

The first time any user logs in to the Administrator Console they are prompted to set their email address. If the user is the cloud administrator they will receive cloud alert emails at this address along with new account requests. Non-administrative users will also receive emails to this address regarding any new account requests that they have submitted. Users are also prompted to change their initial password to a value that only they know.

Administrator Console Overview

Once logged in, you will have access to the Administrator Console.



The header area includes the logo, the link to a user profile setting menu, and the big search box.

The Quick Links area provides links to various contents of the Administrator Console. It is organized into sections made up of two levels. The top level is a heading for that section. Under each heading is a second section that contains a list of links.

The center part of the main screen displays the main panel, usually the search results list. In many content displays, the Administrator Console displays a toolbar that contains action buttons. The bottom of the content area provides the page navigation controls.

The bar at the bottom of the main screen shows system status messages, log window toggle button and the software version (from left to right).

Administrator Tools

The Eucalyptus administrator tools are command-line equivalents of the Eucalyptus Administrator Console functionality. In some cases, they actually offer more functionality than the Eucalyptus Administrator Console. The are automatically installed on the Cloud Controller in the `/usr/sbin` directory. They include commands such as:

- `euca_conf`

- `euca-describe-services`
- `euca-modify-services`
- `euca-describe-arbitrators`
- `euca-modify-properties`
- `euca-register-<*>`
- `euca-deregister-<*>`
- and others

Euca2ools Management

Euca2ools are a set of command-line user tools that can manage Eucalyptus objects including images, instances, security groups, IP addresses, keypairs, volumes, and so on. Euca2ools are located in the `/usr/bin` directory.

You can read about available options for each of the euca2ools commands by using one of the following methods:

```
# <euca2ools_command> -h
# <euca2ools_command> --help
# man <euca2ools_command>
```

Note:

Not all euca2ools commands support all three command help methods.

Euca2ools Syntax

Euca2ools emulate the command-line tools distributed by Amazon (api-tools and ami-tools) and generally accept the same command-line options and honor the same environment variables.

Commands must be authenticated using command-line arguments or environment variables set by the `eucarc` file. Argument syntax:

```
<euca2ools_command> <auth_args> <command_specific_args>
```

If the `eucarc` file has been read by the user's shell, authentication arguments do not need to be provided. If you do need to provide these manually, here are the list of authentication arguments and their definitions:

- **-a, --accesskey**: User's access key ID
- **-s, --secretkey**: User's secret key
- **-c, --cert**: Path to the user PEM-encoded certificate
- **-k, --privatekey**: Path to the user PEM-encoded private key
- **-U, --url**: URL of the cloud to connect to
- **--ec2cert**: Path to the cloud X509 public key certificate

The command-specific arguments will vary by command. To find the available options for any command, refer to the command's help pages.

Third-Party Tools

Eucalyptus has over 200 partners.

Note: For a complete list, see the Eucalyptus partner page at http://www.eucalyptus.com/partners.

Eucalyptus partners are divided into several categories:

- Platform
- System Integrators and Resellers

- Products and Services

Third-party tools are available for configuring and monitoring Eucalyptus private clouds, as well as creating and managing Eucalyptus images.

Lab - Management Tools

In this lab exercise you will log in to the Eucalyptus Administrator Console, perform its first-access configuration, and learn how to navigate its interface.

Lab Objectives:

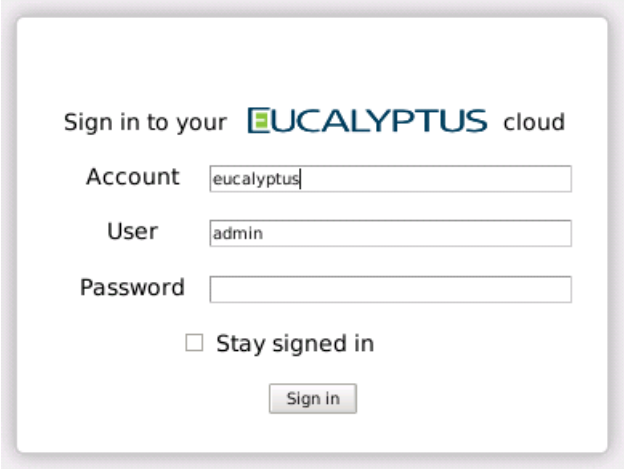
- Perform first-time configuration of the Eucalyptus Administrator Console

Perform First-Time Configuration of the Eucalyptus Administrator Console

Desktop

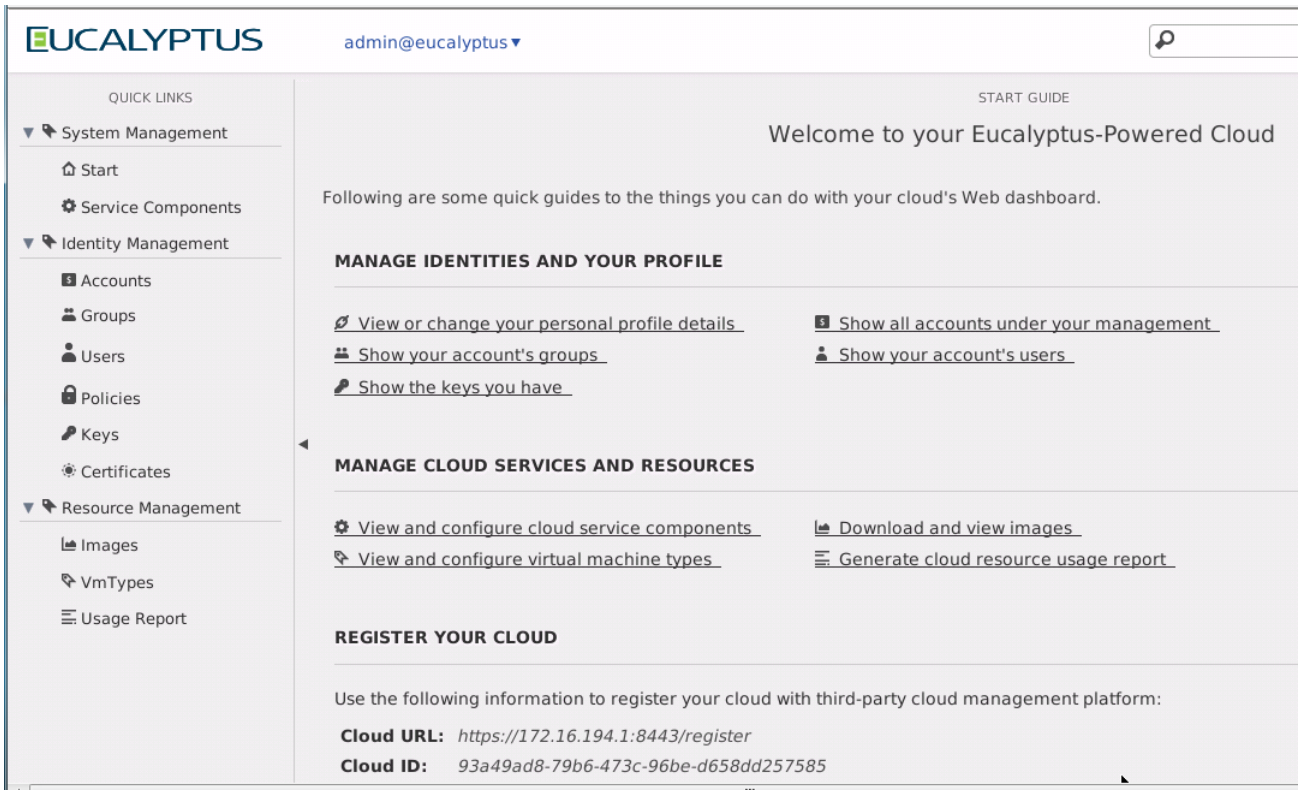
The first time that you log in to the Eucalyptus Administrator Console you will be prompted to perform a few first-time setup tasks. You will perform these first-time tasks in this section of the lab.

1. Open the browser on your Debian desktop and connect to the Administrator Console using the URL https://<front_end_public_IP>:8443. Because the Cloud Controller uses a self-signed certificate, you will need to confirm the security of the connection to the browser. In the login window, sign in to the cloud as the user `admin` in the `eucalyptus` account using the default password of `admin`.



2. When prompted, provide the email address of the *admin* user (for lab purposes, enter any email address), and change the *admin* password to `passwordNN`, where *NN* is the number of your student pod. For example, if you are assigned to pod17, the password would be password17. Save the changes by clicking **OK**.

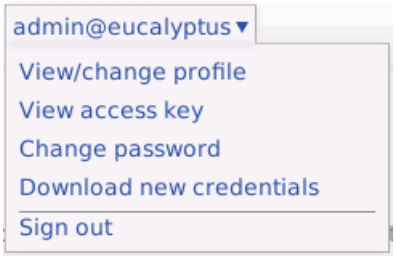
The Administrator Console appears in your browser window.



3. Spend a few minutes exploring the Administrator Console *without making any changes* to your cloud environment.

Note: Later lectures will explain the Administrator Console and later lab exercises will use it to perform tasks in the cloud. Making changes now might inadvertantly break future lab operations.

4. Once you have spent a few minutes looking around in the Adminstrator Console, sign out of it.



5. Close the browser window.

Instance and Image Management

Instance and image management are complex topics that cover a wide range of areas and tasks. We will focus on the following elements of image and instance management:

- Eucalyptus machine images and instances
- Creating, viewing, and deleting key pairs
- Starting, viewing, rebooting, and terminating an instance
- Downloading certified images
- Bundling, uploading, and registering images
- Listing images
- Creating new images
- Downloading and unbundling images
- Deleting images and buckets

Instances Introduction

Working with instances usually involves working with key pairs as well as starting, viewing, rebooting, and terminating an instance.

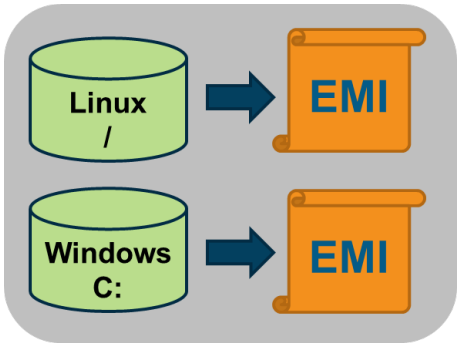
Eucalyptus Machine Images

A Eucalyptus machine image (EMI) is a copy of a virtual machine bootable file system stored in the Walrus storage. Some people find it useful to think of them as virtual machine templates from which multiple identical instances - or copies of the virtual machine - can be deployed.

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

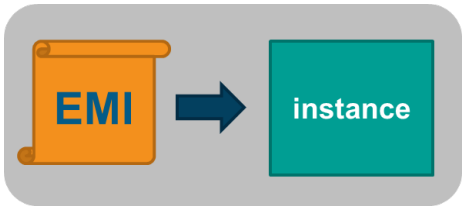
EMIs can be Linux or Windows-based. In Linux it is an image of the / file system while in Windows it is an image of the C: drive.

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



Instances

A virtual machine deployed from an EMI is known as an instance. 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 lesson focuses primarily on instance store-backed instances.



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 was i-12121212, the MAC address of that instance would be D0:0D:12:12:12:12.

Multiple instances can be deployed using a single command-line 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>.

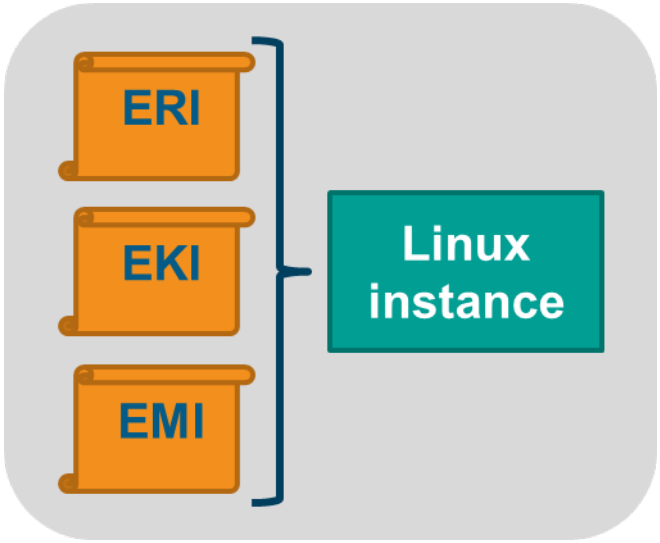
Note: A *reservation* is an EC2 term used to describe the resources reserved for one or more instances launched in the cloud.

Linux and Windows Instances

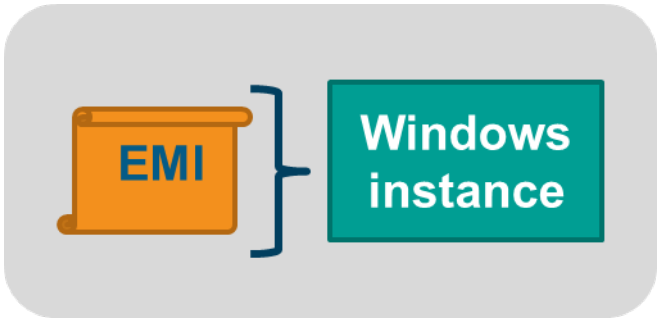
A Linux instance store-backed instance actually consists of three separate images:

- 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.



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*.

Virtual Machine Types (vmtypes)

A virtual machine type, known as a vmtype, 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 vmtypes in Eucalyptus. You can change the quantity of resources associated with each of the five vmtypes, but you cannot change the name of the vmtypes or the number of vmtypes available. If you customize the sizes they must be well-ordered. That means that the CPU, memory, and storage sizes of the next vmtype must be equal to, or larger than, the size of the preceding vmtype.

VIRTUAL MACHINE TYPES

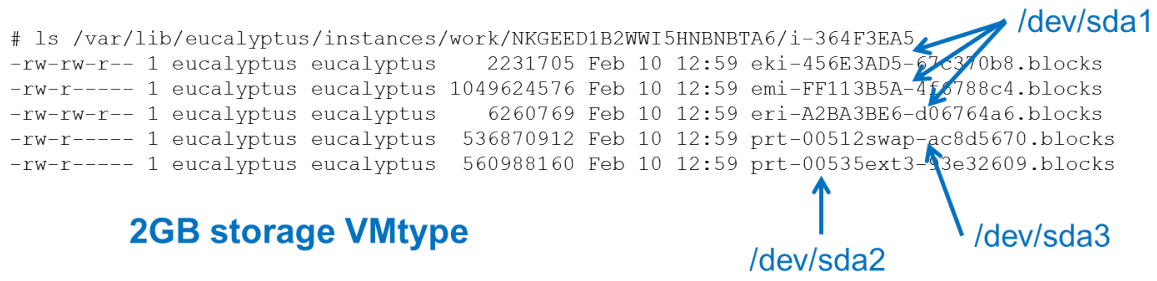
Name	CPUs	Memory (MB)	Disk (GB)
m1.small	1	512	5
c1.medium	2	512	10
m1.large	2	1024	15
m1.xlarge	2	2048	20
c1.xlarge	4	4096	20

The vmtype used to instantiate an EMI must have a defined disk size larger then the EMI file. If a 6GB EMI is loaded into an instance with a vmtype 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 vmtype that is too small, they will recieve an on-screen warning and the operation will terminate.

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.

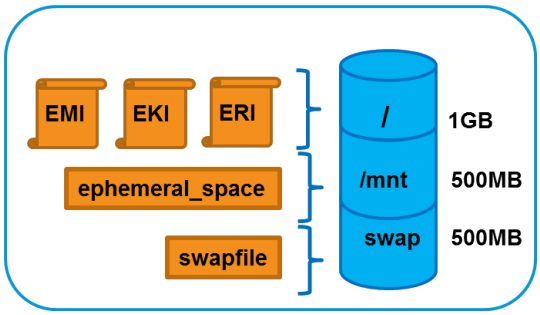
As an instance store-backed instance is launched, several files are brought together using loop devices on the Node Controller. As these files are brought together they form what looks like a disk to the instance's operating system. The illustration below lists a some of the files that make up a running 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.



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`.

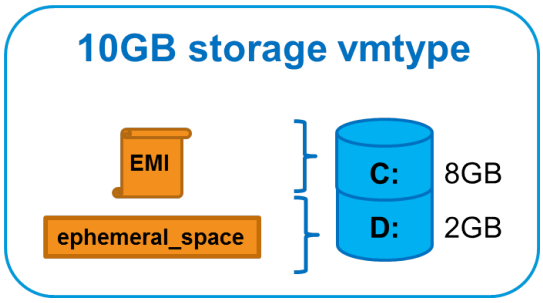
This means the EKI, ERI, EMI, and swap files have consumed approximately 1.5GB of the available 2GB of storage space. The remaining 500GB is allocated to the file with the string `ext3` in its name. In our example, this space is formatted as an ext3 file system and is made available to the instance as 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

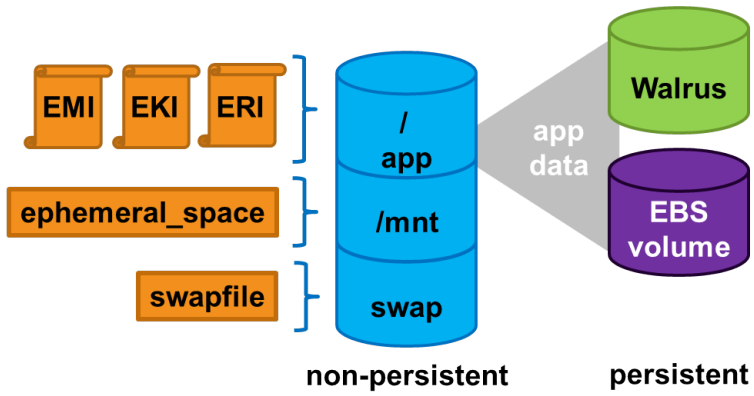
It is also possible to launch ephemeral Windows instances. Just like their Linux counterparts, modifications to Windows instance store-backed instances are lost when instances are purposely or accidentally terminated.

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.



Persistence in Ephemeral Instances

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 Walurs using HTTP put/get operations. Walrus storage is also persistent.



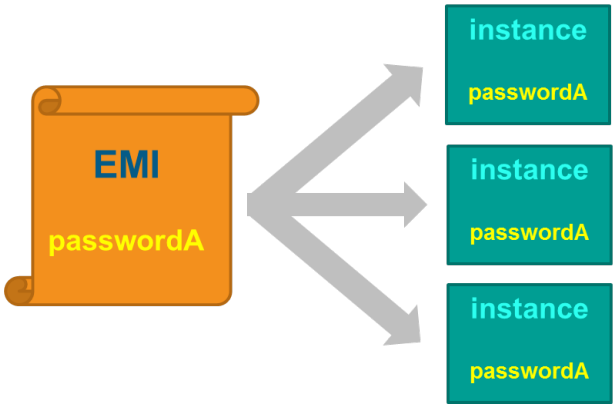
If the application cannot be rewritten to send data to a volume or the Walrus, then the application should be deployed inside an EBS-backed instance. EBS-backed instances are persistent and operate in a manner more similar to a physical machine. EBS-backed instances are covered in another section of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

Using Key Pairs

Setting a preconfigured administrative password in image that can be used by multiple users across an organization is usually viewed as a security risk. To address this, log in access to a running Linux instance is typically controlled by public/private key authentication. Log in to a running Windows instance is accomplished using the normal RDP process, but under the control of a public/private key pair.

Instance Log In Without a Password

Log in access to a running instance is typically accomplished, or least affected by, by public/private key authentication. The problem with including a root (Linux) or Administrator (Windows) password in an EMI is that all users who deploy an instance from that EMI would receive the same administrative password. The result is that one user would know the administrative password for another user's instance. For this reason, the EMIs do not typically include any passwords for the administrative account (root or Administrator). The use of public/private key pairs for each user eliminates the need to include a root or Administrator password in the EMI.



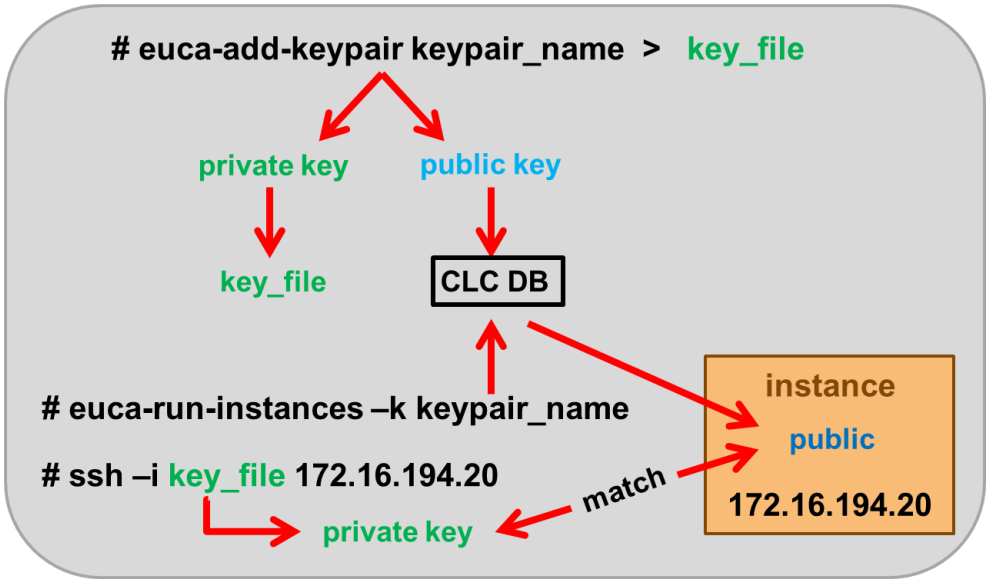
Key Pair in Action - Linux

To log in to a Linux instance that was launched using a key pair, use the following command:

```
# ssh -i <key_file> <ip_address>
```

The diagram below illustrates how public/private key authentication works during log in to a Linux instance. First, the `euca-add-keypair` command creates a public/private key pair for the user. This key pair is known by a key pair name chosen by the user. The private key is placed in a file and stored where ever the user decides to store their key files, their home directory for example. The public key file is stored in the Cloud Controller database and can be referenced using the key pair name.

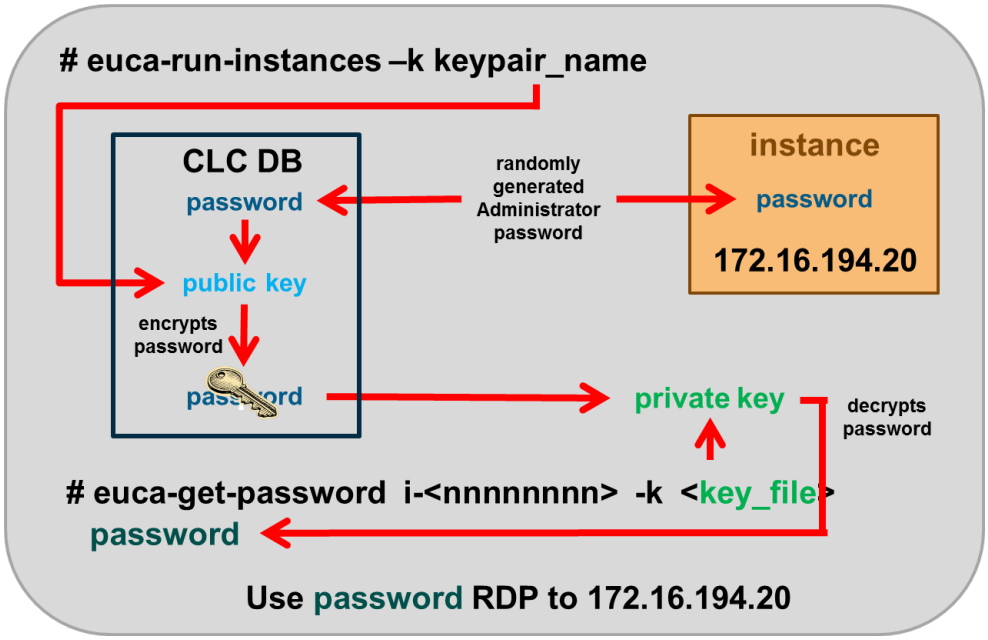
When the user later launches a Linux instance, they specify the name of the key pair that will be used to authenticate log in attempts to the instance. The instance's start up scripts are configured to download the user's public key from the Cloud Controller database during boot up. When the user then uses Secure Shell to log in to the instance, they specify the name of the file containing the matching private key. If the private key matches the public key, the user is granted access.



Key Pair in Action - Windows

Windows instances do not support Secure Shell log in and instead require Remote Display Protocol (RDP). However, RDP requires a user name and password in order to allow login access. The standard login name is Administrator but there is no way ahead of time for the user to know the Administrator password.

Eucalyptus resolves this by choosing a random Administrator password and encrypting it with the public key of the user's key pair. To discover the password, the user runs the `euca-get-password` command with the ID of the instance and the name of the key pair. The encrypted password is fetched from the Cloud Controller, decrypted using the user's private key, and displayed on the screen. The user can then use this password to log in to the Windows instance using RDP.



The Eucalyptus Windows Integration software, installed in the Windows EMI, allows the instance to connect to an existing Active Directory (AD) domain. This allows a user to log in to the instance as Administrator using `euca-get-password`, or log in as a normal user who is included in the AD database. The Eucalyptus Windows Integration software also has a utility to add permissions for users or groups to use RDP to the Windows desktop. (By default, only the Administrator can use RDP.) For more information about creating and EMI with the Eucalyptus Windows Integration software, see the *Eucalyptus Users Guide* at *http://www.eucalyptus.com/docs*.

Managing Key Pairs - Euca2ools

Before launching an instance create a public/private key pair using the following command:

```
euca-add-keypair <keypair_name> | tee <key_file>
```

Piping to the `tee` command while generating a key file is optional, but is useful because you will immediately see if any errors occur. The key will be written both the screen and to the file if the command is executed successfully. The `<keypair_name>` is a human-friendly name chosen by the user. For increased security change the permissions on the key file by using the follow command:

```
chmod 600 <key_file>
```

In fact, the private key file will be ignored by SSH without this permission change.

You can view key pairs with the following command:

```
euca-describe-keypairs <verbose>
```

Note: The `<verbose>` option can only be used by the cloud administrator. It displays all key pairs and not just those owned by the cloud administrator.

It is possible for a user to have more than one key pair, but if too many key pairs exist then management of those key pairs can become difficult. To delete a key pair use the following command:

```
euca-delete-keypair <keypair_name>
```

There is an important distinction to keep in mind. There are keys and certificates that authenticate a user to the cloud and keys that authenticate a user to a running instance. The key pairs described in this section are used only to authenticate a user to an instance. Other keys and certificates that identify a user to the cloud are used to allow a user to run euca2ools and other cloud-interface tools.

Start an Instance - Euca2ools

To start an instance using euca2ools, run the following command:

```
euca-run-instances -k <keypair_name> --kernel eki-<nnnnnnnn> /
--ramdisk eri-<nnnnnnnn> emi-<nnnnnnnn>
```

This command starts the instance in their account's *default* security group. The user's public key specified by `-k <keypair_name>` is inserted into the running instance for authentication purposes. The EMI will be launched with the specific kernel and ramdisk images specified on the command line. This will override any other default kernel or ramdisk images that might apply.

To specify a security group other than the account's *default*, add the `-g <group_name>` option to the command.

To start an instance in a specific availability zone (cluster), add the `-z <cluster_name>` option to the command.

To start multiple instances, add `-n <count>` to the command.

To start an instance with only a private IP address, add the `--addressing private` option to the command. Without this option, instances running in a cloud in MANAGED or MANAGED-NOVLAN modes will receive both a private and public IP address, assuming that there are available public IP addresses remaining in the pool.

Kernel and Ramdisk Association

There are three ways to associate an EMI with a kernel image and ramdisk image.

- Allow the instance to run with the cloud-wide default kernel and ramdisk images. By default, these are the first kernel and ramdisk images registered with the cloud, but this can be changed by a cloud administrator.

  Note: An incorrect kernel or ramdisk image can cause a Linux instance to fail to boot.

- Bundle the EMI with specific kernel and ramdisk images. These become the defaults for this specific image. This can be accomplished using the following command:

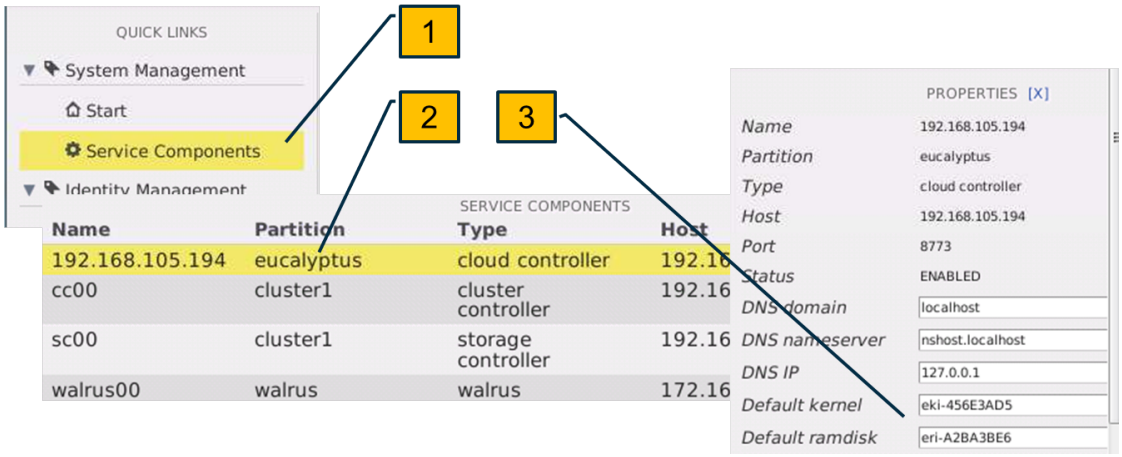
  ```
  euca-bundle-image -i <boot_image> --kernel /
     eki-<nnnnnnnn> --ramdisk eri-<nnnnnnnn>
  ```

- Associate the EMI with a specific kernel and ramdisk at runtime. This overrides any default images that exist. To accomplish this from the command line, use the following syntax:

  ```
  euca-run-instances -k <key_name> --kernel /
     eki-<nnnnnnnn> --ramdisk eri-<nnnnnnnn> /
     emi-<nnnnnnnn>
  ```

Default Kernel and Ramdisk Changes

The cloud-wide default kernel and ramdisk images can be changed using the Eucalyptus Administrator Console or the command line. To change the default images from the Eucalyptus Administrator Console:



To change the default kernel and ramdisk images from the command line:

```
euca-modify-property –p cloud.images.defaultkernelid=“”
euca-modify-property –p cloud.images.defaultramdiskid=“”
```

Listing Instances - Euca2ools

To view running instances using euca2ools, use the following command:

```
# euca-describe-instances <verbose>
```

Note: The `<verbose>` option can only be used by the cloud administrator. It displays all instances and not just those owned by the cloud administrator.

The output from this command will show you a number of different pieces of information:

```
# euca-describe-instances
RESERVATION     r-DF17402D      362593875553    default
INSTANCE        i-B6B03DB9      emi-F059361E    172.16.194.22   10.110.195.88
running  andrew-test     0    c1.medium       2012-11-16T20:52:20.417Z
cluster1        eki-E0C13DEE    eri-6FCF3A23    monitoring-disabled
172.16.194.22    10.110.195.88
```

The fields are defined as follows:

- Reservation ID,
- User ID who launched the instance
- Security group name
- Instance ID formatted as i-<nnnnnnnn>
- The EMI that the instance is based on
- Public and private IP addresses of the instance
- Instance's status
- Key pair name
- Index number - if muliple instances are launched at once, they will get unique index numbers start with 0 and incrementing with each instance
- vmtype
- Date and time the instance was launched

- Availability zone (cluster)
- Kernel image the instance is based on
- Ramdisk image the instance is based on
- Monitoring state
- Public and private IP addresses

Stopping an Instance - Euca2ools

An instance store or EBS-backed instance can be rebooted or terminated. Data stored in an ephemeral instance is not lost during a reboot, but is lost when an instance is terminated.

Note: Some versions of hypervisors might not reconnect a volume to the instance after a reboot and manual intervention might be necessary.

To reboot an instance use the following command:

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

To terminate an instance use the following command:

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

Images Introduction

There are a few different types of images that you will need to manage. Management of those images will include tasks such as downloading certified images from Eucalyptus, bundling, uploading, and registering images, viewing images, creating new images or modifying existing images, downloading and unbundling images, deregistering images, and deleting images.

Bundle an Image

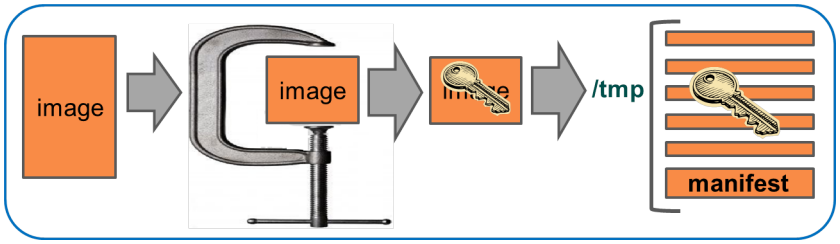
Initially an image is a `.img` file that contains the bootable file system needed to launch a virtual machine in the cloud. However, images (`.img` files) must be bundled to be used in a Eucalyptus cloud. Bundling prepares the `.img` file to run in the Eucalyptus cloud. To bundle an image, run the following command:

```
euca-bundle-image -i /<path>/<image_file>
```

Bundled images are stored on your local machine in the `/tmp` by default, but can be redirected to another location by using the `-d <directory>` option. This can be useful, for example, if you do not have enough space in `/tmp`.

By default, images are bundled as 64-bit images, but you can force 32-bit by using the `-r i386` option.

During the bundling process, the image is compressed to minimize network bandwidth usage and storage space. The compressed image is then encrypted with the user's credentials and signed to ensure confidentiality of the data when the image is uploaded over the network from the local machine to the Walrus. Next the encrypted image is split into manageable parts for later upload. Finally, an XML manifest file is created containing a list of the image parts with their checksums. The result is a bundled image.

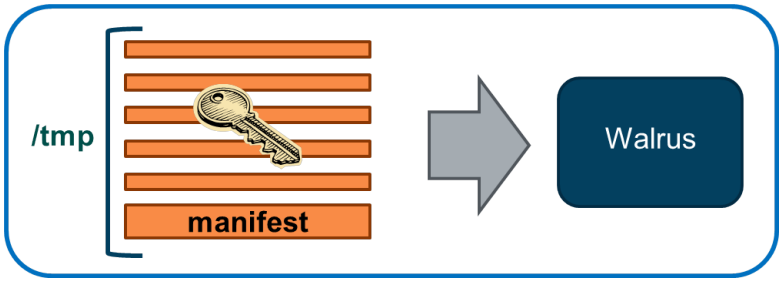


Upload a Bundled Image

The bundled image needs to be uploaded to a Walrus bucket. To do this, run the following command:

```
euca-upload-bundle -b <bucket> -m </path_to_manifest_file>
```

Any bucket name can be specified, and that bucket will be created if it does not already exist. The bundled image is uploaded to `/var/lib/eucalyptus/bukkit/<bucket_name>` on the Walrus host.



There are at least two possible schemes for using and organizing buckets:

- Create separate buckets for kernel images, ramdisk images, and bootable file system images
- Create separate buckets for each type of operating system

Register an Uploaded Bundle

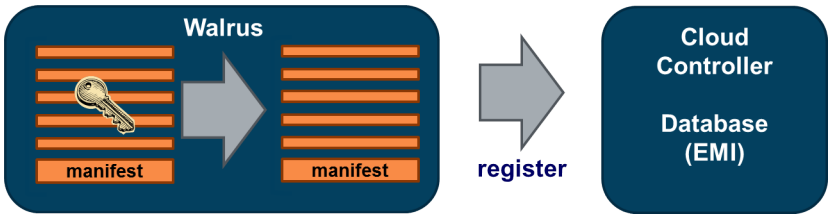
Uploaded bundles must be registered with the Cloud Controller before they can be used to launch instances.

Registering a bundle decrypts it using the user's credentials.

Note: Only a cloud administrator can register kernel and ramdisk bundles.

To register an bundle, run the following command:

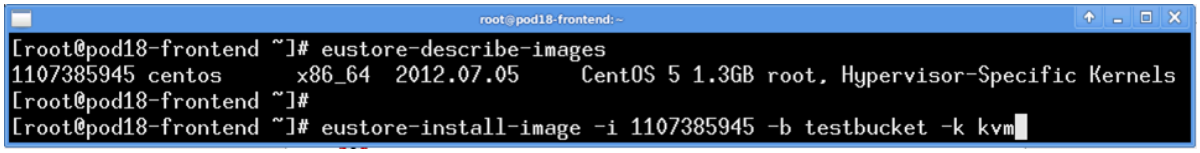
```
euca-register <bucket>/<manifest_file>
```



Download an Experimental Image

Eucalyptus provides experimental Linux images for proof-of-concept testing, or for whatever purpose a user might find them useful. These images are stored in the Eucalyptus Store, or EuStore for short. It is expected that the list of experimental images in the EuStore will grow over time.

To list the images in the EuStore, use the following `eustore-describe-images` command. To download and install an image from the EuStore to your cloud use the `eustore-install-image` command.



Note: Since the time of this screen capture, additional images have been added to the Eustore.

The `eustore-install-image` command:

- Downloads the bootable file system, kernel, and ramdisk image files to the local system
- Bundles (prepares) the images for upload to Walrus
- Uploads the bundles to Walrus
- Registers the bundles as EMI, ERI, and EKI images

List Images - Euca2ools

You can list available images registered with the Cloud Controller by using the `euca-describe-images` command. The output should look something like this:

```
# euca-describe-images
IMAGE   emi-FF113B5A    centos/centos.5-3.x86-64.img.manifest.xml
714937189257    available       public i386     machine eki-456E3AD5
eri-A2BA3BE6    instance-store
IMAGE   eri-A2BA3BE6    centos/initrd-2.6.27.21-0.1-xen.manifest.xml
714937189257    available       public i386     ramdisk
instance-store
IMAGE   eki-456E3AD5    centos/vmlinuz-2.6.27.21-0.1-xen.manifest.xml
714937189257    available       public i386     kernel
instance-store
```

Creating New Images

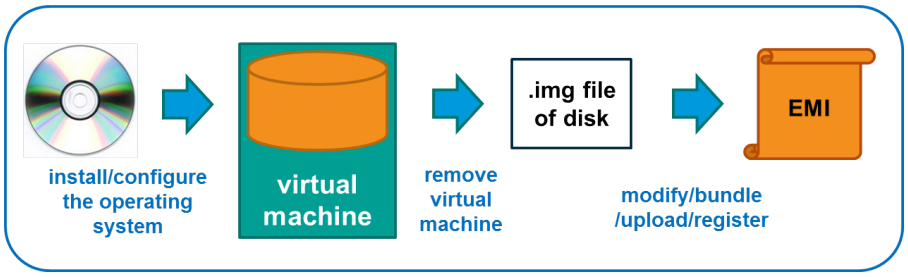
There are multiple techniques to create a new image.

- Create a brand-new image file from installation media
- Modify an existing image file
- Create a new image from a running system - either physical or virtual
- User a third-party software tool

Create a New Image from Installation Media

To create a brand-new image from installation media is a time-consuming, multi-step, and potentially error-prone process. One of the ways to accomplish this is to log in to the Node Controller and manually create a virtual machine using KVM or libvirt utilities. The virtual machine should have a virtual disk file large enough to hold the operating

system and any applications you plan on installing. Once the virtual machine exists, use installation media to install the operating system and any applications on the virtual disk. The virtual disk will actually be a `.img` file on the Node Controller.



Once you have a virtual machine disk file (an `.img` file), you will still need modify the image file before you bundle, upload, and register the image. Unfortunately, errors that occur early in the image creation process can go undetected until a user tries to use the image to launch a new instance. For more information about manually creating new image files, see the *Eucalyptus User Guide* at *http://www.eucalyptus.com/docs*.

Because this method can be time-consuming, software tools exist and are being developed that are designed to build images for you and remove some of the possibility of human error.

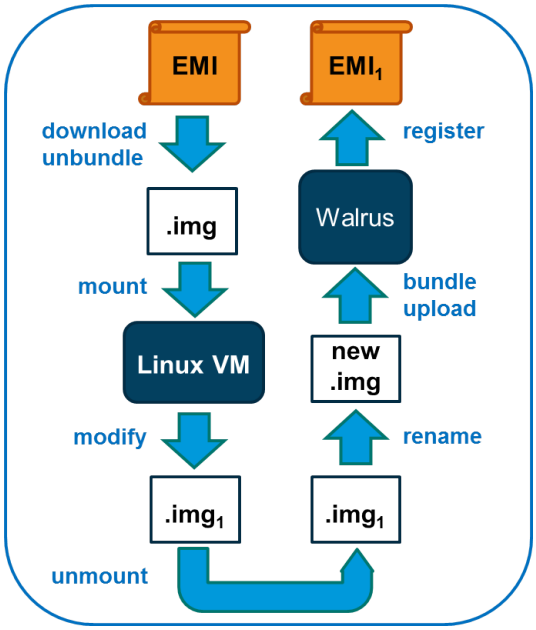
Note: Examples of such tools include AMI Creator (*https://github.com/katzj/ami-creator*), BoxGrinder (*http://boxgrinder.org*) and UForge (*http://www.usharesoft.com*).

In order to simplify image creation and management in the cloud, many administrators will provide users with basic, known working images and allow them to modify these existing images to create new ones. This is often a simpler and faster approach.

Modify a Linux Image File

If a Linux image already exists that can be modified to meet the needs of a new instance, starting from this image can dramatically reduce the amount of time and potential error in image creation. To modify an existing Linux image without launching it as an instance:

1. If necessary, download and unbundle an existing image.
2. Mount the image to a Linux host using loopback mounts.
3. Modify files and directories in the image's file system.
4. Unmount the image.
5. Save the image to a new file name.
6. Bundle, upload, and register the new image.



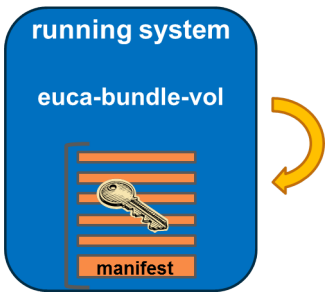
Modify a Running Linux System

Alternatively, you can make changes to a running Linux system and then use the following command to create an image of the running system. The only requirement is that the following command be present on the running Linux system, which requires that euca2ools be installed on it.

```
euca-bundle-vol <args>
```

There are quite a number of different options for the `euca-bundle-vol` command and which options are included depend on the situation. The actual steps used to bundle a running Linux system will also vary with each Linux distribution and version.

The image will be created on the hard drive of the running Linux system so the free space available to bundle the image should be twice that of the current disk space used by the running system.



Once the running system's image has been bundled on the running system, you would then run `euca-upload-bundle` and `euca-register` to transfer the image to the Walrus and register it with the Cloud Controller.

Modify a Running Windows Instance

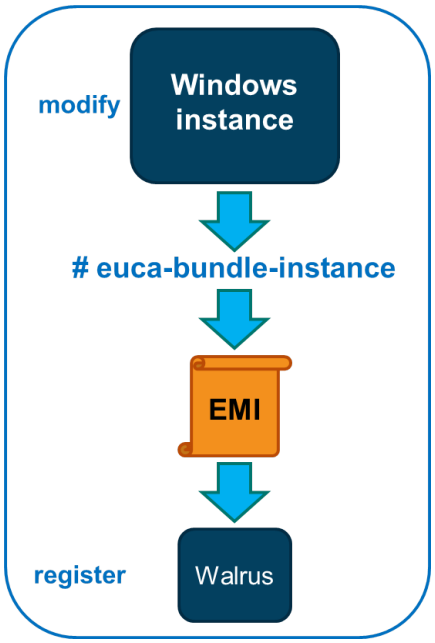
A Windows image can only be modified from within a running instance. To do this:

1. Start a Windows instance.
2. Modify the running instance as needed.

3. Bundle the running instance and upload it to Walrus.

To bundle a new Windows image from a running instance:

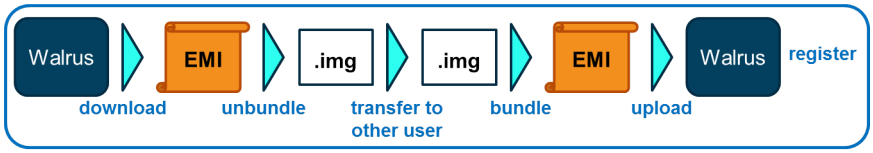
```
euca-bundle-instance -b <bucket_name> i-<nnnnnnnn>
```



The image is automatically uploaded to the Walrus so there is no need to run the `euca-upload-bundle` command. You only need to register the uploaded bundle with the Cloud Controller.

Download and Unbundle an Image

Images created by the *admin* user are, by default, available to all other cloud users. Images created, bundled, uploaded, and registered by normal users, by default, are not available to other users in the cloud. One method to allow an existing image created by one user to be shared by other users is to first download and unbundle it on a local system. Once it is available on a local system, the user can then send it to another user where they re-bundle, upload, and register it with their credentials.



To download a bundled image from Walrus, run the following command:

```
euca-download-bundle -b <bucket> -m <manifest> /
   -d <local_directory>
```

Once downloaded, a bundled image can be unbundled using the following command:

```
euca-unbundle -m <manifest> -s <source_directory> /
   -d <destination_directory>
```

The image can now be delivered to a new user, who can bundle, upload, and register it using their credentials.

Deregister an Image

Before an image is deleted from Walrus, you should always deregister the image from the Cloud Controller. To accomplish this run the following command:

```
euca-deregister <image_ID>
```

Running the command a single time deregisters the image but leaves it visible when you run `euca-describe-images`. However, the image is listed as *deregistered*. The fact that information about the image is still visible after deregistration is useful in that it displays the bucket name and manifest file name. These two pieces of information are required in the command that deletes an image.

Running the `euca-deregister` command a second time removes it from the `euca-describe-images` output.

Delete an Image

Once deregistered, a single image can be deleted from the Walrus. An entire bucket can also be deleted from the Walrus.

To delete a specific image, first run `euca-describe-images` to view the bucket name and manifest file name. To delete the image, run the following command:

```
euca-delete-bundle -b <bucket> -p <manifest_file_prefix>
```

The manifest file prefix is everything in the manifest file name up to but not including the `.manifest.xml` suffix.

To delete an entire bucket, run the command without specifying a manifest prefix. Use the `--clear` option instead:

```
euca-delete-bundle -b <bucket> --clear
```

Note: **Caution**: Remember to deregister images before deleting them!

Lab - Instance and Image Management

In this lab exercise you will download and install a CentOS 5 Linux image from the EuStore. You will create a key pair that can be used to authenticate to an instance at log in. Using the CentOS image you will launch, view, and terminate instances using the euca2ools. You will also customize the image in order to make a new image. Lastly, you will deregister and remove an image from your cloud.

Lab Objectives:

- Download and register a CentOS 5 image
- Launch and connect to an instance using euca2ools
- Download and unbundle an image
- Customize an image
- Test the customized image
- Remove an image

Download and Register a CentOS 5 Image

Eucalyptus provides Linux images that can be downloaded to help test a newly installed Eucalyptus cloud. They are available for download using the Eucalyptus image store, or EuStore. The EuStore is accessed using the `eustore-*` command-line commands.

1. Desktop If necessary, from the Debian desktop open an SSH session to the front-end host.

```
# ssh <front_end_public_IP>
```

2. Front End List the images available in the EuStore.

```
# eustore-describe-images
```

3. Front End Download and register the 1.3GB x86_64 CentOS image, ID 1107385945, from the EuStore. Place it in a Walrus bucket named *centos* and make sure to choose the KVM kernel. This command will take several minutes to complete its tasks.

```
# eustore-install-image –i 1107385945 –b centos –k kvm
```

4. Front End Verify that you can see three registered images in your cloud; an EKI, ERI, and EMI.

```
# euca-describe-images
```

5. Front End Open TCP port 22 (for SSH) in the *default* security group before trying to connect to the console of an instance. (This command, along with security groups, will be discussed in a later module.)

```
# euca-authorize –P tcp –p 22 –s 0.0.0.0/0 default
```

Launch and connect to an instance using euca2ools

In this section of the lab you will use the command-line euca2ools to view images, create a key pair, launch an instance, log in to an instance, and terminate an instance.

1. Desktop On the Desktop, open an SSH session to the front-end host if an SSH session is not currently open.

```
# ssh <front_end_public_IP>
```

2. Front End On the front-end host, view the registered images.

```
# euca-describe-images
```

3. Front End On the front-end host, create a key pair named *adminkey* that can be used to launch and then authenticate to an instance during log in and then view the available key pair .

```
# euca-add-keypair adminkey > ~/.euca/adminkey.priv
# euca-describe-keypairs
```

4. Front End On the front-end host, launch a new instance using your key pair name and the available EMI.

```
# euca-run-instances –k <keypair_name> emi-<nnnnnnnn>
```

5. Front End On the front-end host, view the instance.

```
# euca-describe-instances
```

Allow the instance to transition from a `pending` to a `running` state. Run the command multiple times until you see the instance reach a `running` state. Note the public IP address assigned to the instance.

6. Front End Once the instance reaches a running state, use SSH on the front-end host to connect to the instance's console. You will need to access the private key file you copied to `/root` earlier in the lab. You can list the key file in the `/root` directory.

```
# cd /root
# ls
# ssh –i <key_file> <public_IP_of_instance>
```

7. Front End Type `exit` to disconnect from the instance.

```
# exit
```

8. Front End On the front-end host, terminate the instance. You will need the instance ID to accomplish this task.

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

Download and unbundle an image

Front End

It is possible to recreate an image from a bundle of an image. To do this you must first download the bundle from Walrus and then use the bundle as the source to create the new image. This is useful if you want to modify an existing image in the cloud in order to create a new image.

1. On the front-end host, view the available images.

```
# euca-describe-images
```

2. In order to download a bundle from Walrus, you first need to determine which file system on your front-end host has sufficient disk space to hold the downloaded and unbundled image. Use the `df -h` command to view available disk space in your file systems.

```
# df -h
```

Note: Notice that `/var` has a large amount of disk space available.

3. Create a directory beneath `/var` to hold bundled images downloaded from Walrus.

```
# mkdir /var/downloads
```

4. Download the CentOS images in the *centos* bucket to the `/var/downloads` directory on the front-end host.

```
# euca-download-bundle –b centos –d /var/downloads
```

5. Recreate the original root file system image from the downloaded EMI bundle. Place the image file in `/var/images`.

```
# cd /var/downloads
# ls
# mkdir /var/images
# euca-unbundle -m euca-centos-5.8-2012.07.05-x86_64.manifest.xml -d /var/
images
```

6. View the recreated root file system image file. It should be approximately 1.4GB in size.

```
# ls –l /var/images
```

Customize an image

Because most Eucalyptus users require custom instances, image management plays a key role in Eucalyptus administration. Custom images are typically based on a preferred version of a preferred operating system with a set of required applications pre-installed. In this section of the lab you will customize an image using an existing image as a starting point. You will modify a Eucalyptus-provided CentOS 5 image by changing its configuration and installing additional software in it. Once the image has been modified you will save it under a new name and register it as a new image in the cloud.

1. On the front-end host, locate the CentOS root file system image (`euca-centos-5.8-2012.07.05-x86_64.img`) that you downloaded and unbundled in the previous lab section. It should be in the directory `/var/images`.

```
# cd /var/images
# ls
```

2. Because you will modify the CentOS image, rename it now. Change the name from `euca-centos-5.8-2012.07.05-x86_64.img` to `kvm-centos-5.8.img`.

```
# mv euca-centos-5.8-2012.07.05-x86_64.img kvm-centos-5.8.img
# ls
```

3. Create a directory named `/var/tempmnt` to use as a file system mount point.

```
# mkdir /var/tempmnt
```

4. Access the root file system in the image (`.img`) file by using a Linux loop device. A loop device allows you to mount and use an image file as though it were a file system mounted on a disk device. Configure a loopback mount to mount the CentOS root file system image file to the mount point you created earlier, and then verify that it was mounted.

```
# mount -o loop kvm-centos-5.8.img /var/tempmnt
# mount
```

5. Repair the mount points listed in the image's `/etc/fstab` file. Although you have been able to launch instances, they have not actually been booting correctly up to this point. The `/etc/fstab` file currently instructs the Linux operating system to mount `/`, `/mnt`, and the swap area from the devices `/dev/sda1`, `/dev/sda2`, and `/dev/sda3`. Because your instances run under a KVM hypervisor and with the VIRTIO paravirtualized device drivers enabled in the `eucalyptus.conf` file, your instances should instead access these file systems and the swap area from the devices `/dev/vda1`, `/dev/vda2`, and `/dev/vda3`. Use an editor (either `vi` or `nano –w`) to modify the device names in the image's `/etc/fstab` file. Remember, the image's `/etc/fstab` file is currently mounted at `/mnt/extra/tempmnt/etc/fstab`.

```
# vi /var/tempmnt/etc/fstab

/dev/vda1    /           ext3    defaults,errors=remount-ro 0 0
/dev/vda2    /mnt        ext3    defaults                   0 0
/dev/vda3    swap        swap    defaults                   0 0
devpts       /dev/pts    devpts  gid=5,mode=620             0 0
proc         /proc       proc    defaults                   0 0
```

6. Edit the image's `/etc/rc.local` file for editing using either `vi` or `nano –w`. Remember, the image's `/etc/rc.local` file is currently mounted at `/var/tempmnt/etc/rc.local`. You will add several lines of commands to the end of this file over the next several steps in the lab.

```
# vi /var/tempmnt/etc/rc.local
```

7. Move to the end of the file and add the `yum` command to remove the old version of euca2ools that is pre-installed in the instance. Leave the file open to continue editing.

```
yum –y remove euca2ools
```

8. Add the command sequence to download and install the NTP RPM package. Euca2ools will not operate correctly if the instance is not time synchronized with the Cloud Controller. Leave the file open to continue editing.

```
yum -y install ntp
```

9. Next add the command to install the euca2ools repo RPM package. The repo file provides euca2ools software download information to the `yum` installer. Leave the file open to continue editing.

```
rpm –Uvh http://downloads.eucalyptus.com/software/euca2ools/2.1/centos/5 \
/x86_64/euca2ools-release-2.1.noarch.rpm
```

10. Now add the `yum` command to install euca2ools using the information available in the euca2ools repo file. Save your changes to the image's `/etc/rc.local` file and exit the editor before moving to the next step in the lab.

```
yum –y install euca2ools
```

11. Euca2ools are already installed in the image, but the euca2ools must be able to authenticate to the Cloud Controller. To automate this authentication, configure the image to include the `.euca` directory with the cloud administrator's credentials. Copy the credentials of the user *admin* from the front-end host to the image file.

```
# cp -r /root/.euca /var/tempmnt/root
```

12. Use an editor (`vi` or `nano –w`) to create root's `.bashrc` file in the image. Add the command to read and execute the commands in the `eucarc` file.

```
# vi /var/tempmnt/root/.bashrc

source /root/.euca/eucarc
```

13. Unmount the image file and verify that it was unmounted.

```
# cd
# umount /var/tempmnt
# mount
```

Test the customized image

In this section of the lab exercise you will bundle, upload, and register the new customized image. Then you will launch an instance from the image, synchronize its time using NTP, run a euca2ools command, and check that the new `/etc/fstab` file is working correctly.

1. On the front-end host, bundle, upload, and register your new image. Bundle the new EMI with the existing EKI and ERI that appear in the output of the `euca-describe-images` command. Upload the bundle to a new bucket named *kvm-centos*. It will take a minute or two for the bundling of the image to complete.

```
# cd /var/images
# ls
# euca-describe-images
# euca-bundle-image -i kvm-centos-5.8.img --kernel eki-<nnnnnnnn> \
--ramdisk eri-<nnnnnnnn>
# euca-upload-bundle -b kvm-centos -m /tmp/kvm-centos-5.8.img.manifest.xml
# euca-register kvm-centos/kvm-centos-5.8.img.manifest.xml
```

2. View your new EMI. It should be the one in the *kvm-centos* bucket.

```
# euca-describe-images
```

3. Launch a new instance using the EMI of your newly customized CentOS image. It will be the EMI in the *kvm-centos* bucket.

```
# euca-describe-keypairs
# euca-run-instances –k <keypair_name> emi-<nnnnnnnn>
```

4. Monitor the instance until it reaches a running state. It might take 1-2 minutes longer to boot as it is de-installing and reinstalling software as it boots.

```
# euca-describe-instances
```

5. Use SSH to log in to the new instance.

```
# cd /root
# ls
# ssh –i <key_file> <instance_public_IP>
```

6. On the instance, update the instance's time with NTP.

```
# ntpdate pool.ntp.org
```

7. Source the `.bashrc` file and verify that the `eucarc` file environment variables were set.

```
# source .bashrc
# env | grep EC2
```

8. Test the operation of euca2ools by viewing the public IP address pool in your cloud.

```
# euca-describe-addresses
```

9. Log out of the instance but leave it running.

```
# exit
```

10. On the front-end host, launch a second instance using the original EMI. It will be the EMI in the *centos* bucket. Monitor it until it reaches a running state.

```
# euca-describe-images
# euca-run-instances –k <keypair_name> emi-<nnnnnnnn>
# euca-describe-instances
```

11. Use SSH to log in to the instance you just launched using the original EMI image in the *centos* bucket.

```
# cd /root
# ls
# ssh –i <key_file> <instance_public_IP>
```

12. In the instance, view the mounted file systems and the available swap space using the `df –h` and `swapon –s` commands.

```
# df –h
# swapon -s
```

Which file systems are mounted? Which swap area is available? Write them down.

13. Log out of the instance.

```
# exit
```

14. Use SSH to log back in to the new instance launched from the EMI in the *kvm-centos* bucket.

```
# ssh –i <key_file> <instance_public_IP>
```

15. In the instance, view the mounted file systems and the available swap space using the `df –h` and `swapon –s` commands.

```
# df –h
```

```
# swapon -s
```

Which file systems are mounted? Which swap area is available? Write them down. How do they compare to the instance launched from the original EMI. What is missing? This is why you modified the image's `/etc/fstab` file earlier in this lab.

16. Log out of the instance.

```
# exit
```

17. On the front-end host, terminate both instances.

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

Remove an image from a Walrus bucket

In this section of the lab you will unregister and delete an EMI image from a Walrus bucket.

WARNING! Be careful to deregister images before removing them from the Walrus.

1. On the front-end host, view the registered images.

```
# euca-describe-images
```

2. Deregister the EMI image in the *centos* bucket. This is the image with the incorrect `/etc/fstab` file and no installed NTP package.

```
# euca-deregister emi-<nnnnnnnn>
```

3. Run the `euca-describe-images` command again. What changed in the output?

```
# euca-describe-images
```

What changed in the output?

4. Remove the deregistered EMI from the *centos* bucket. The `-p` prefix option should include everything in the manifest name up to, but not including, `.manifest.xml`.

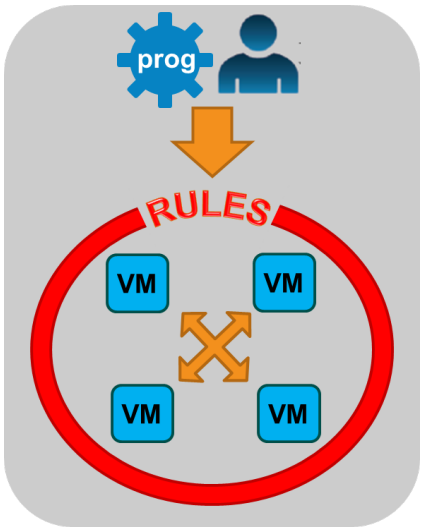
```
# euca-delete-bundle -b centos -p euca-centos-5.8-2012.07.05-x86_64
```

5. Remove the EMI entry by running the `euca-deregister` command again.

```
# euca-deregister emi-<nnnnnnnn>
# euca-describe-images
```

Security Groups

A security group is a set of network access rules applied to all instances associated with the group. They only control ingress (inbound) network traffic. Network traffic between the instances within a security group is unrestricted. Security groups are implemented on the Cluster Controller by `iptables`' *nat* and *filter* tables. Eucalyptus automatically updates `iptables` based on user firewall rules as instances are launched and terminated.



At startup, Eucalyptus flushes any existing `iptables` firewall rules. Then Eucalyptus modifies only the `iptables`' *filter* and *nat* tables, as necessary, to protect and provide access to running instances. If you have existing firewall rules that you would like to preserve at Eucalyptus startup, enter them in `iptables` before Eucalyptus is started and then run the command `iptables-save > /var/run/eucalyptus/net/iptables-preload`. Your rules will be added to the `iptables-preload` file. When Eucalyptus starts, it will add the rules in this file to the Eucalyptus firewall configuration.

Note: To view changes in `iptables`, you can run the following commands on the Cluster Controller:

```
iptables -L
iptables -t nat -L
service iptables status
```

Security groups are available in MANAGED and MANAGED-NOVLAN network modes.

Default Security Group

Each account, not each user, is given a *default* security group. For example, if there were five accounts in your cloud there would be five security groups named *default*. Unless directed to do otherwise, all instances launched by users in an account are assigned to the account's *default* security group.

Initially all security groups deny ingress (inbound) network traffic from all sources. Users can open access to specific ports and protocols from specific networks or open access to other security groups.

Users can create and use their own security groups. All user-created security groups are listed with the user's account as the owner. However, only the user that created the security group can modify or delete the security group, which maintains the security of security groups.

If a user has created a new security group, they can run an instance in that security group from the command line using the following command:

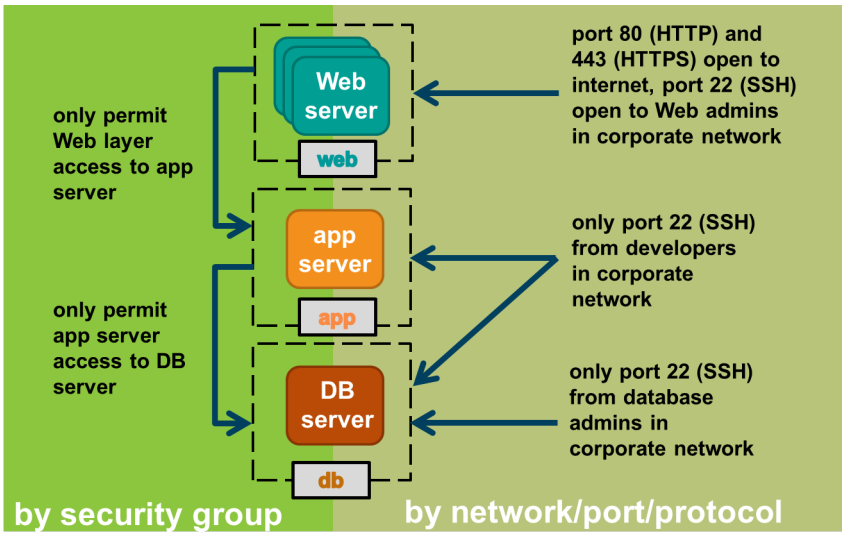
```
euca-run-instances -g <group_name>
```

Running this command without the `-g <group_name>` option will run the instance in the *default* security group.

Users can also specific a specific security group when launching an instance using the Eucalyptus User Console.

Security Groups Example

Here is an example of security group design that controls access to a multi-tiered application. Each tier in the application is protected by its own security group. Each security group controls connection requests from users outside of the cloud as well as connection requests from the other security groups within the cloud.



The *web* security group allows users outside of the cloud to access the Web servers using TCP ports 80 and 443 This provides these users access to the Web-based application. Web administrators are allowed access to the Web servers through Secure Shell at TCP port 22. All other ports are blocked.

The *app* security group allows Secure Shell access from application administrators through TCP port 22. It also permits access from the Web server instances in the *web* security group. All other ports are blocked.

Finally, the *db* security group allows Secure Shell access from database administrators through TCP port 22. It also permits access from the application server instances in the *app* security group. All other ports are blocked.

Security Group Management

Security groups are managed using either the command-line euca2ools or the Eucalyptus User Console.

Only the user who created the security group can add or revoke network access rules. Not even a cloud administrator can modify or revoke another user's access rules.

Security Groups - Euca2ools

There are a number of command-line euca2ools available to manage security groups:

- List security groups and their attributes:

```
euca-describe-groups
```

- Add a new security group:

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

- Add a new rule to a security group:

```
euca-authorize -P <protocol> -p <port> /
   -s <network> <group_name>
```

Note: You can specify a range of numbers for the port, for example 80-8080.

Note: Here's an example that would enable ICMP:

```
euca-authorize -P icmp -s 192.168.1.1 /
   -t -1:-1 <group_name>
```

Note: Instead of a CIDR address, you can specify another group that has access:

```
euca-authorize --source-group <group_name> /
    -P <protocol> -p <port> <group_name>
```

- Remove a rule from a security group:

```
euca-revoke -P <protocol> -p <port> /
   -s <network> <group_name>
```

- Delete a security group:

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

Note: If you delete a security group with an instance still running, the security group is maintained until the instance is terminated. If you used the command line or Eucalyptus User Console to list security groups, nothing would be displayed even though the security group still exists.

Lab - Manage Security Groups

In this lab exercise you will create, manage, and use security groups using euca2ools. A security group is a set of firewall rules configured on the Cluster Controller and applied to all instances that are members of the group. Security groups allow Eucalyptus users to control network access to their instances. Security groups are available in the MANAGED or MANAGED-NOVLAN network modes.

Lab Objectives:

- Create a security group using euca2ools
- Modify a security group using euca2ools
- Delete a securitty group using euca2ools

Create a security group using euca2ools

In this section of the lab exercise you will create a security group using euca2ools. Then you will add a firewall rule to your security group.

1. Desktop If necessary, from the Debian desktop open an SSH session to the front-end host.

```
# ssh <front_end_public_IP>
```

2. Front End Add a new security group. Name the new group *mygroup2* .

```
# euca-add-group -d "another security group" mygroup2
```

3. Front End Display the available security groups and the network access rules associated with each group.

```
# euca-describe-groups
```

Are there any permissions associated with your new security group?

4. Front End Add a rule to your new security group that allows SSH access from any network.

```
# euca-authorize -P tcp -p 22 -s 0.0.0.0/0 mygroup2
```

5. Front End Verify that the rule was added to your security group.

```
# euca-describe-groups
```

6. Front End Launch a new instance in your *mygroup2* security group. Wait for it to enter a running state.

```
# euca-describe-images
# euca-run-instances –k <keypair_name> –g mygroup2 emi-<nnnnnnnn>
# euca-describe-instances
```

7. Front End Log in to your instance to verify that the new firewall rule is working. Exit out of your instance when you are finished.

```
# cd /root
# ls
# ssh -i <key_file> <instance_public_IP>
# exit
```

Modify a security group using euca2ools

In this section of the lab exercise you will use euca2ools to modify the firewall rules associated with a security group.

1. Front End On the front-end host, add a new rule to the *mygroup2* security group that allows ICMP *echo requests*.

```
# euca-authorize –P icmp –t -1:-1 –s 0.0.0.0/0 mygroup2
```

2. Front End Verify that the rule was added to your security group.

```
# euca-describe-groups
```

3. Desktop From the Debian desktop, open a new xterm window and `ping` the public IP address of your running instance.

```
# ping <instance_public_IP>
```

Did it work? It should have.

4. Desktop Close the xterm window on the Debian desktop (where you ran the `ping` command).
5. Front End On the front-end host, remove the rule from the *mygroup2* security group that allows ICMP connections.

```
# euca-revoke –P icmp –t -1:-1 –s 0.0.0.0/0 mygroup2
```

6. Front End On the front-end host, verify that the rule was removed from your security group.

```
# euca-describe-groups
```

Delete a security group using euca2ools

Front End

In this section of the lab exercise you will delete a security group using euca2ools.

1. On the front-end host, delete the *mygroup2* security group.

```
# euca-delete-group mygroup2
```

You should have received an error message because the security group is being used by a running instance.

2. On the front-end host, terminate the running instance.

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

3. On the front-end host, delete the *mygroup2* security group.

```
# euca-delete-group mygroup2
```

Did the command work this time?

4. On the front-end host, verify that the *mygroup2* security group was deleted.

```
# euca-describe-groups
```

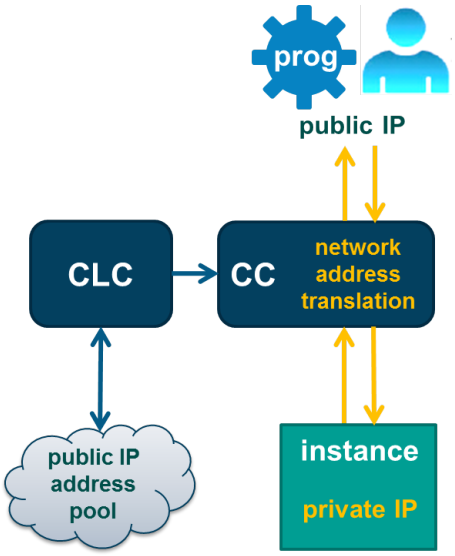
Elastic IP Addresses

This section of the course discusses public IP addresses and elastic IP addresses in detail.

Public IP Addresses

Public IP addresses are available in MANAGED and MANAGED-NOVLAN network modes. There are assigned to an instance so that the instance is reachable from outside of the cloud.

The Cluster Controller assigns the public IP address to the instance, but under the control of the Cloud Controller. The Cloud Controller ensures that the same public IP address is not assigned to two different instances by two different Cluster Controllers.



Elastic IP Introduction

An elastic IP address is just a public IP address that a user manually assigns to an instance. It replaces a cloud-assigned public IP address. Elastic IP addresses are available when running in MANAGED and MANAGED-NOVLAN network modes.

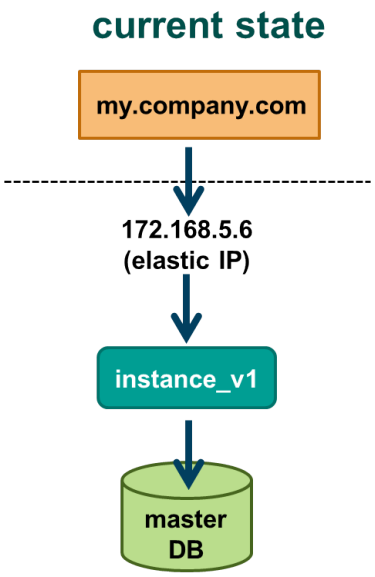
Elastic IP addresses allow a user to assign a service running in an instance a well-known, static IP address - for example, a Web server.

To use an elastic IP address, a user reserves an IP address from the pool of available public IP addresses. This address is reserved until the user releases the IP address back to the pool. The user then assigns the reserved IP address to a running instance. The original cloud-assigned public IP address is released back into the pool. If the instance is terminated, the elastic IP address remains reserved by the user but is available for assignment to another instance.

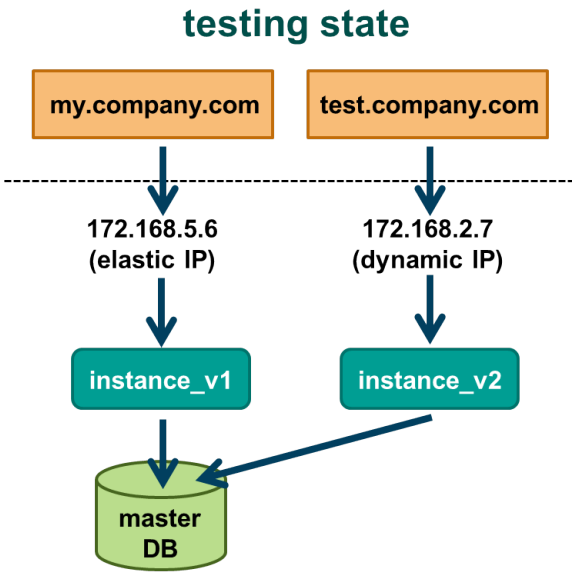
The user is not able to reserve a specific address from the public pool. When a user requests an address they receive the next available public IP address in the pool.

Elastic IP Address Example

For example, assume that Susan has a CentOS 5.x Web server running at my.company.com and she has configured it with an elastic IP address. The actual IP configuration might look something like this:

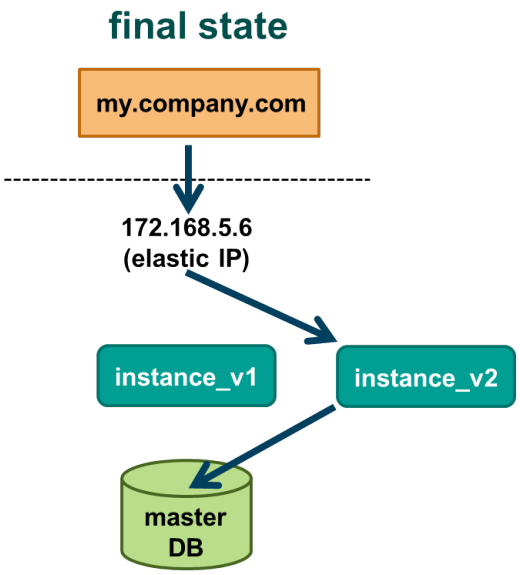


Assume that Susan wants to upgrade the Web server to CentOS 6.x. First, she would launch the new Web server as a new instance in the cloud and test it to ensure that it is working properly. During testing the current Web server's operations are unaffected.



Once she is satisfied that everything is working as it should on the new Web server, she would then re-map the elastic IP address from the old Web server to the new Web server.

Note: No changes to public DNS are required to make this change. This means that existing DNS nameserver caches that might exist around the intranet and Internet still contain valid address information.



However if something were to go wrong at this point, Susan could re-map the elastic IP address back to the old Web server and the change would happen instantaneously. If however, everything continues to work as expected, Susan could decommission the old Web server and the upgrade would be complete.

Manage Elastic IP Addresses - Euca2ools

There are a number of command-line euca2ools available to work with elastic IP addresses.

- View public IP addresses:

```
euca-describe-addresses <verbose>
```

Note: The `<verbose>` option can only be used by the cloud administrator. It displays all addresses and not just those owned by the cloud administrator.

- Reserve an elastic IP address from the public IP address pool:

```
euca-allocate-address
```

- Assign an elastic IP address to an instance:

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

- Disassociate an elastic IP address from an instance:

```
euca-disassociate-address <IP>
```

- Release an elastic IP address back to the public address pool:

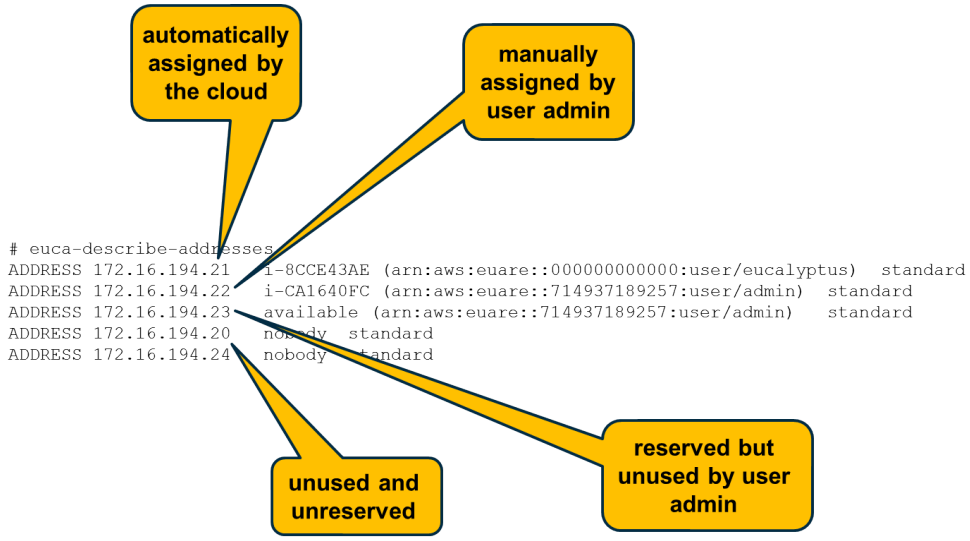
```
euca-release-address <IP>
```

Viewing IP Addresses

When you run `euca-describe-addresses` to view the pool of public IP addresses, you might see a number of different entry types.

If the IP address is followed by an instance ID, that IP address has been assigned to that instance. Following the instance ID you will see the user that assigned the IP address to the instance. Elastic IP addresses will have a user name listed, whereas a cloud-assigned public IP address will have *eucalyptus* listed.

IP addresses not currently assigned to instances may show up as *available* or *nobody*. An *available* status indicates that the IP address has been reserved by a user but has not yet been assigned to an instance. This will be followed by the name of the user who reserved the address. A *nobody* status indicates that the address is available for either reservation or automatic assignment by the cloud when a new instance is launched.



Lab - Manage Elastic IP Addresses

In this lab exercise you will configure, manage, and use elastic IP addresses using euca2ools. An elastic IP address is a public (external to the cloud) IP address that users can reserve and, on the fly, associate with a specific instance. Elastic IP addresses are available when Eucalyptus is running in MANAGED or MANAGED-NOVLAN network mode. You will also learn to launch an instance that does not use a public IP address. This is useful in situations where instances must be able to communicate with each other, but do not need to communicate with nodes outside of the cloud.

Lab Objectives:

- Reserve an elastic IP address using euca2ools
- Assign an elastic IP address using euca2ools
- Unassign an elastic IP address using euca2ools
- Release an IP elastic address using euca2ools
- Configure and test private-only IP addressing

Reserve an elastic IP address using euca2ools

In this section of the lab you will use euca2ools to view public IP addresses and reserve a public IP address from the pool of public IP addresses available in your cloud.

1. Desktop From the Debian desktop, if necessary, use SSH to log in to the front-end host.
2. Front End From the front-end host, launch a new instance.

```
# euca-describe-images
# euca-run-instances -k <keypair_name> emi-<nnnnnnnn>
```

3. Front End View the public IP address of your running instance.

```
# euca-describe-instances
```

4. Front End View the public IP addresses that are available.

```
# euca-describe-addresses
```

Note that *eucalyptus* assigned the public IP address to your instance.

5. Front End Reserve a public IP address.

```
# euca-allocate-address
```

6. Front End View the reserved IP address.

```
# euca-describe-addresses
```

Note that the user *admin* has reserved an IP address.

Assign an elastic IP address using euca2ools

Front End

In this section of the lab you will use euca2ools to assign a reserved public IP address to a specific instance.

1. With your instance running, assign the reserved IP address to your instance.

```
# euca-associate-address -i i-<nnnnnnnn> <reserved_IP>
```

2. View the public IP addresses again. Note that the IP address assigned to your instance was assigned by the user *admin* and not by *eucalyptus*.

```
# euca-describe-addresses
```

3. View your running instance again. Note that the IP address assigned to your instance has changed.

```
# euca-describe-instances
```

Unassign an elastic IP address using euca2ools

Front End

In this section of the lab you will use euca2ools to disassociate a reserved public IP address from a specific instance.

1. Disassociate the reserved IP address from your instance.

```
# euca-disassociate-address <allocated_IP>
```

2. View the public IP addresses again.

```
# euca-describe-addresses
```

Note that the IP address assigned to your instance was unassigned and is available for assignment.

3. View your running instance again.

```
# euca-describe-instances
```

Note that the IP address assigned to your instance has changed.

Release an elastic IP address using euca2ools

In this section of the lab you will use euca2ools to release a reserved IP address back to the pool of public IP addresses available in your cloud.

1. Release the reserved but unused (available) IP address.

```
# euca-release-address  <reserved_available_IP>
```

2. View which addresses are reserved by the user *admin*.

```
# euca-describe-addresses
```

3. Terminate your instance.

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

Configure private IP addressing

In this section of the lab you will use euca2ools to launch an instance that is not assigned a public IP address.

1. 

On the front-end host, launch an instance using the `--addressing private` option.

```
# euca-describe-images
# euca-run-instances –k <keypair_name> --addressing private emi-<nnnnnnnn>
```

2. 

View your running instance again.

```
# euca-describe-instances
```

Note the public IP address field. What address does it contain?

3. Desktop On the Debian desktop, open an xterm window and then use Secure Copy to copy the *admin's* private key file from `/root` on the front-end host to `/root` on the Debian desktop.

```
# scp <public_IP_of_front-end>:/root/<key_file> /root/<key_file>
```

Leave the xterm window open.

4. Desktop On the Debian desktop in the xterm window, use SSH to log in to the instance using the private IP address.

```
# ssh –i <key_file> <private_IP_of_instance>
```

The Debian desktop is outside of the cloud. Did the log in work? Leave the xterm window open.

5. Front End On the front-end host, launch another instance and allow it to have a public IP address. Wait for it to achieve a running status.

```
# euca-describe-images
# euca-run-instances –k <keypair_name> emi-<nnnnnnnn>
# euca-describe-instances
```

6. Desktop On the Debian desktop in the open xterm window, use SSH to log in to the new instance.

```
# ssh –i <key_file> <public_IP_of_new_instance>
```

7. Desktop While logged in to the new instance, use Secure Copy to copy *admin's* keypair key file from `/root` on the front-end host to `/root` on the instance.

```
# scp <public_IP_of_front-end>:/root/<key_file> /root/<key_file>
```

8. Desktop While logged in to the new instance, use SSH to log in to the instance that has only a private IP address. The new instance is both inside the cloud and inside the same security group.

```
# ssh –i <key_file> <private_IP_of_instance>
```

Did it work this time?

9. Desktop In the xterm window, log out of the instance that has only the private IP address.

```
# exit
```

10. Desktop In the xterm window, log out of the instance with the public IP address.

```
# exit
```

11. Desktop Close the xterm window after you have logged out of the instance.

12. Front End On the front-end host, terminate both instances.

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

Volumes and Snapshots

This module provides information about three primary topics:

- Volume management
- Snapshot management
- Boot-from-EBS instances

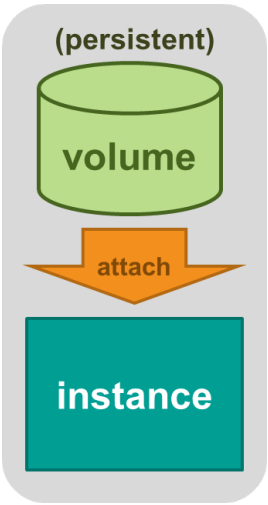
Eucalyptus Block Store

Eucalyptus enables you to create persistent storage volumes called Eucalyptus block volumes. This is the Eucalyptus version of Amazon Elastic Block Storage (EBS). Volumes are mounted as SCSI devices by instances. For example, in a cloud running the KVM hypervisor a volume in a Linux instance appears as an *vd* device in the `/dev` directory.

Note: The use of a *vd* device assumes that VIRTIO disk paravirtualization enabled, otherwise *sd* devices appear in the `/dev` directory.

Once a volume is attached to an instance, it behaves like a raw, unformatted block device. You can create a file system on a volume or use a volume in any other way you would use a block device.

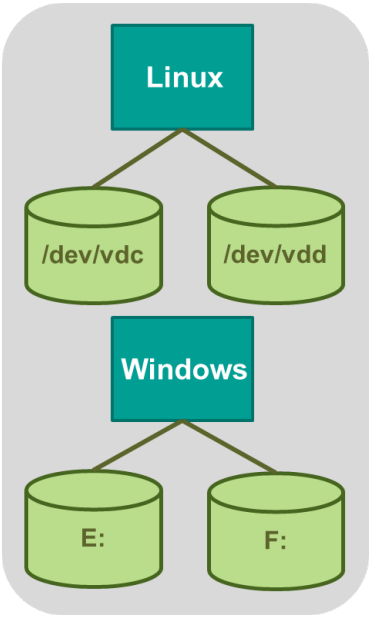
A volume is a per-cluster resource. A volume is available only within an individual cluster and is not available across clusters.



Volume Access

Volumes are managed by the Storage Controller and are assigned a unique ID in the format vol-<nnnnnnnn>. Only a single instance can access a volume at one time, however, an instance can mount multiple volumes.

Volumes are attached to instances as SCSI disks. In Linux, they are available as disk devices in the `/dev` directory. A Windows instance will mount a volume as a SCSI device and you typically would initialize it and assign it a logical drive letter.



Note: A Linux Node Controller is limited to a total of 64 volumes for all instances. An ESXi host is limited to a total of 14 volumes for all instances.

Manage Volumes - Euca2ools

Euca2ools provides a number of command-line tools for managing volumes:

- To create a new volume:

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

- To create a volume from an existing snapshot:

```
euca-create-volume --snapshot vol-<nnnnnnnn> -z <cluster_name>
```

- To view the list of volumes:

```
euca-describe-volumes <verbose>
```

Note: The `<verbose>` option can only be used by the cloud administrator. It displays all volumes and not just those owned by the cloud administrator.

- To attach a volume to an instance:

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

- To detach a volume from an instance:

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

- To delete a volume:

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

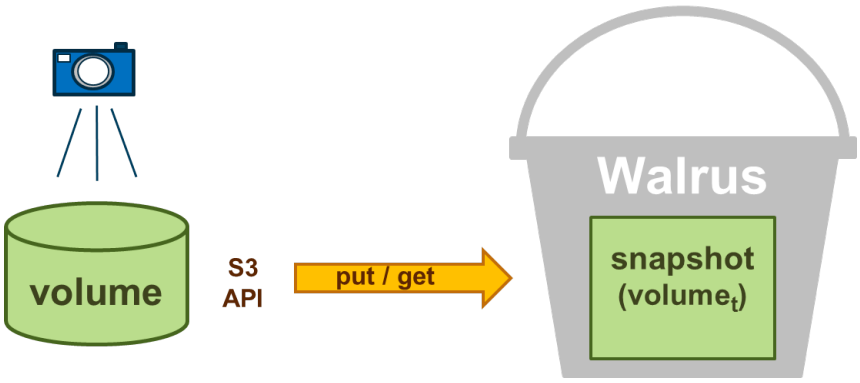
Once a volume is attached to a Linux instance, it would have to be partitioned, have a file system created, and then mounted. Once a volume is attached to a Windows instance, it would have to be initialized, partitioned, and assigned a drive letter.

Volume Snapshots

Eucalyptus provides the ability to create point-in-time snapshots of volumes. Each snapshot of a volume is assigned a unique ID in the format snap-<nnnnnnnn>, where <nnnnnnnn> is a hexadecimal number.

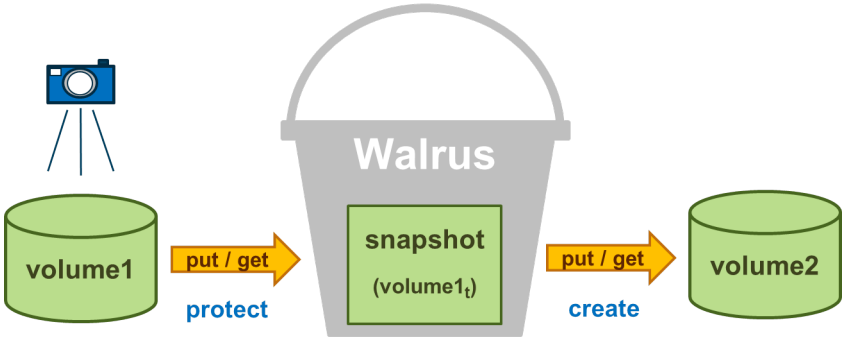
Note: Depending on the application, you might have manually flush pending data writes cached in memory to the volume before creating a snapshot. This helps to ensure application data consistency in the snapshot.

Snapshots are created on the Node Controller, but are automatically transferred to, and cached on, the Storage Controller. From the Storage Controller they are transferred to the Walrus. On the Walrus, snapshots are persistently stored at `/var/lib/eucalyptus/volumes/snap-<nnnnnnnn>`.



Using Snapshots

Snapshots can be used to protect data for long-term durability, as well as create new volumes.



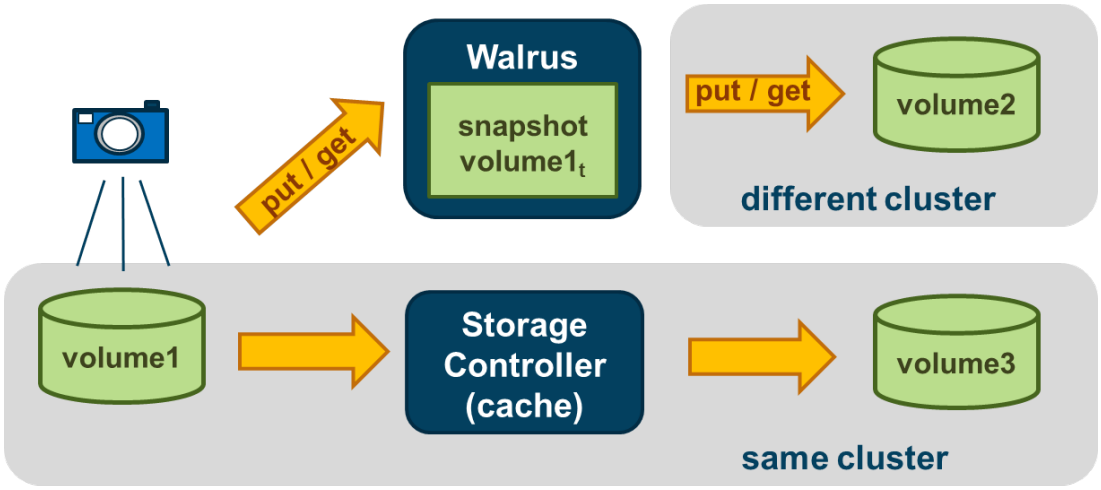
Create a Volume from a Snapshot

A snapshot can be used to create a new volume in a cluster. When a snapshot is taken of a volume, the Storage Controller caches the snapshot before transferring it to the Walrus. If a new volume is created from that snapshot, the

Storage Controller can use its cache rather than have to download the snapshot from the Walrus. This saves time and network bandwidth. If the new volume to be created is located in another cluster, then the Storage Controller in that cluster must download the snapshot from the Walrus.

The snapshot is cached on the Storage Controller at `/var/lib/eucalyptus/volumes/snap-<nnnnnnnn>`.

The diagram below illustrates new volume creation using either the Storage Controller's local cache or using a snapshot downloaded from the Walrus.



Manage Snapshots - Euca2ools

Euca2ools provides several command line tools to manage snapshots, including:

- Create a snapshot:

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

- View snapshots:

```
euca-describe-snapshots <verbose>
```

Note: The `<verbose>` option can only be used by the cloud administrator. It displays all snapshots and not just those owned by the user.

- Delete a snapshot:

```
euca-delete-snapshot snap-<nnnnnnnn>
```

Lab - Manage Volumes and Snapshots

In this lab exercise you will create a Eucalyptus storage volume and attach it to an instance using euca2ools. You will customize data on the volume, take a snapshot of the volume, and create a new volume from the snapshot. You will launch a new instance and attach the new volume to it and view the customized data.

- Create a storage volume using euca2ools
- Attach a volume to an instance using euca2ools
- Partition, mount, and add data to a volume
- Detach a volume using euca2ools

- Snapshot a volume using euca2ools
- Create a volume from a snapshot using euca2ools
- Delete a volume and snapshot using euca2ools

Create a storage volume using euca2ools

In this section of the lab you will create a storage volume using euca2ools.

1. Desktop On the Debian desktop, if necessary use SSH to log in to the front-end host.

```
# ssh <front_end_public_IP>
```

2. Front End In order to create a volume, you will need to know the name of your cluster. On the front-end host, run the following command to display the name of your cluster.

```
# euca-describe-availability-zones
```

3. Front End Create a 2GB volume in your cluster.

```
# euca-create-volume -s 2 -z <your_cluster_name>
```

4. Front End Check the status of the volume until it displays as `available` .

```
# euca-describe-volumes
```

Attach a volume to an instance using euca2ools

Front End

In this section of the lab you will create an instance and then use euca2ools to attach a storage volume to it.

1. Launch an instance and wait for it to reach a running status.

```
# euca-describe-images
# euca-run-instances –k <keypair_name> emi-<nnnnnnnn>
# euca-describe-instances
```

2. Attach the new volume to the instance. Use the block device name `/dev/vdb`.

```
# euca-describe-volumes
# euca-attach-volume -i i-<nnnnnnnn> -d /dev/vdb vol-<nnnnnnnn>
```

3. Check the status of the volume until it displays as `in-use` and `attached`.

```
# euca-describe-volumes
```

Partition, mount, and add data to a volume

Front End

In this section of the lab you will configure the volume so that the instance can use it. This includes creating a partition on the volume, creating a file system on the partition, and then mounting the file system to the Linux operating system. You will finish by writing a file to the file system on the volume.

1. From the front-end host, log in to the instance with the attached volume. Use your previously obtained private key file in root's home directory as the argument to the `ssh` command.

```
# cd /root
# ls
# ssh -i <key_file> <instance_public_IP>
```

2. When you are logged in to your instance, verify that the volume is available to your instance.

```
# fdisk -l /dev/vdb
```

   The `fdisk` command should report the following information about your volume.

```
Disk /dev/vdb doesn't contain a valid partition table
```

3. Create a partition table and a single partition on the volume so that it can be formatted with a file system and then mounted by the operating system. Type the following commands to create primary partition 1 on the volume.

```
# fdisk /dev/vdb
n
p
1
1
(press the Enter key)
w
```

4. Verify that the partition table was created.

```
# fdisk -l /dev/vdb
```

5. Create a Linux ext3 file system on the volume.

```
# mkfs.ext3 /dev/vdb1
```

6. Create the mount point `/data`.

```
# mkdir /data
```

7. Mount the file system.

```
# mount /dev/vdb1 /data
```

8. Verify that the file system is available.

```
# mount
# ls -l /data
```

9. Verify the size of the file system.

```
# df –h /data
```

10. Create a file on the volume so that you can recognize it later.

```
# echo "File attached to instance by admin." > /data/myfile.txt
# cat /data/myfile.txt
```

11. Unmount the file system.

```
# umount /data
```

12. Verify that the file system is not mounted.

```
# mount
```

13. Exit out of the SSH session to the instance.

```
# exit
```

Detach a volume using euca2ools

In this section of the lab you will detach the volume from your instance. You will also terminate the instance that the volume was attached to as the instance will no longer be needed in the lab.

1. From the front-end host, view the volume name.

```
# euca-describe-volumes
```

2. Detach the volume from the instance.

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

3. Terminate the instance as you no longer need it in lab.

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

Snapshot a volume using euca2ools

In this section of the lab you will create a snapshot of your volume. The snapshot will capture the volume “as is”, which means that snapshot will capture your partition information, file system, as well as the file that you wrote to the file system.

1. From the front-end host, view the available volumes.

```
# euca-describe-volumes
```

2. Create a snapshot of the volume.

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

3. Use `euca-describe-snapshots` to check the creation progress of your new snapshot. Note the percentage of progress reported each time you run the command. Wait for a status of `completed`. This operation can take a minute or two.

```
# euca-describe-snapshots
```

Create a volume from a snapshot using euca2ools

Front End

In this section of the lab you will create a new volume using the snapshot taken in the previous section of this lab. The new volume will include your partition information, file system, and the file that you wrote to the file system. You will also create a new instance and attach the new volume to the instance. Then you will view the data on the volume.

1. Create a new volume from the snapshot.

```
# euca-create-volume --snapshot snap-<nnnnnnnn> -z <your_cluster_name>
```

   Note the volume ID as you will need it later.

2. Monitor the progress of volume creation until there is a status of `available` .

```
# euca-describe-volumes
```

   You should have two volumes. Make sure you understand which volume is the new volume. The new volume should have a snapshot ID associated with it.

3. From the front-end host, launch an instance. Wait for it to reach a `running` status.

```
# euca-describe-images
# euca-run-instances –k <keypair_name> emi-<nnnnnnnn>
# euca-describe-instances
```

4. Attach the new volume to the instance. It is the volume associated with the snapshot. Be sure to attach it to the device `/dev/vdb`.

```
# euca-attach-volume -i i-<nnnnnnnn> -d /dev/vdb vol-<nnnnnnnn>
```

5. View the state of the new volume and wait for it to be `attached`.

```
# euca-describe-volumes
```

6. From the front-end host, use SSH to log in to the instance.

```
# cd /root
# ls
# ssh -i <key_file> <instance_public_IP>
```

7. Create the `/data` directory to use as a mount point.

```
# mkdir /data
```

8. Mount the file system located on the volume to the `/data` directory.

```
# mount /dev/vdb1 /data
```

9. Verify that the file system is available and that your customized file is present.

```
# ls -l /data
# cat /data/myfile.txt
```

10. Unmount the file system.

```
# cd /
# umount /data
```

11. Exit the SSH session to the instance.

```
# exit
```

Delete a volume and snapshot using euca2ools

In this section of the lab you will delete a volume and a snapshot using euca2ools. You will also terminate the remaining instance.

1. From the front-end host, view the volume ID number of the attached volume and then detach the volume from the instance.

```
# euca-describe-volumes
# euca-detach-volume vol-<nnnnnnnn>
```

2. List and delete any volumes that you find.

```
# euca-describe-volumes
# euca-delete-volume vol-<nnnnnnnn>
```

3. List and delete any snapshots that you find.

```
# euca-describe-snapshots
# euca-delete-snapshot snap-<nnnnnnnn>
```

4. List and terminate any running instances that you find.

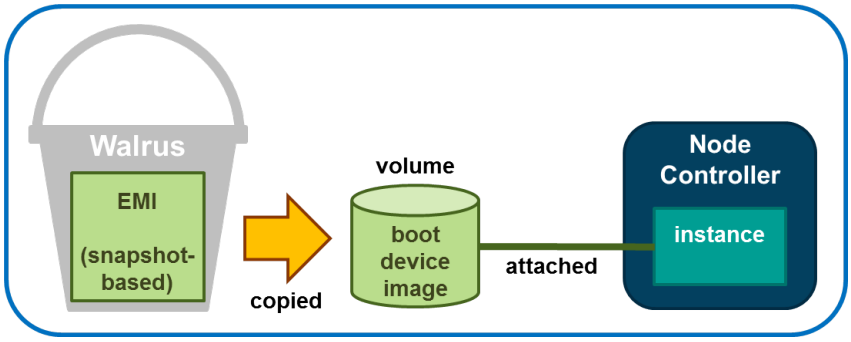
```
# euca-describe-instances
# euca-terminate-instances i-<nnnnnnnn>
```

EBS-Backed Instances

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

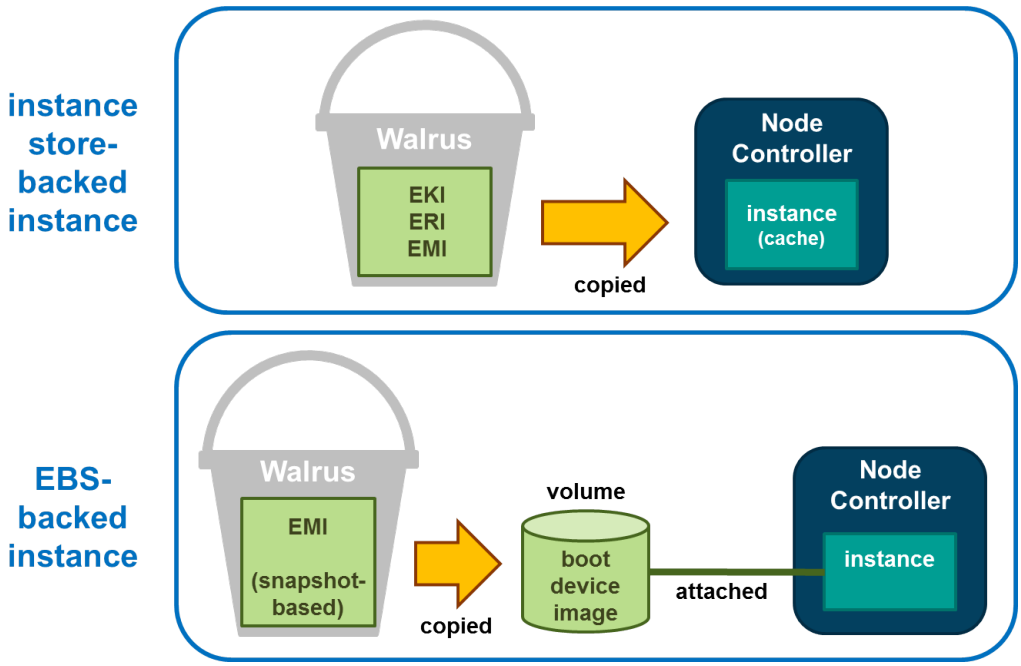
With EBS-backed instances you are booting an instance from a volume rather than a bundled EMI image. The boot volume is created from a snapshot of a root device volume. EBS-backed instances can be either Linux or Windows. The boot volume is persistent so changes to the instance are persistent.

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.

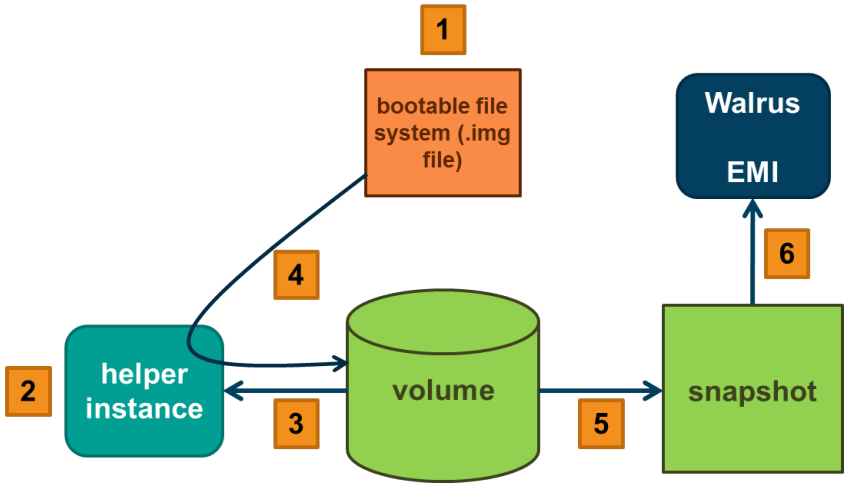
At this point the volume can be detached from the helper instance, a snapshot taken, and the snapshot registered as a new EMI.

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.

Optional Lab - Boot from an EBS Volume

In this optional lab exercise you will create a bootable Eucalyptus volume by copying a preconfigured Linux image file to it. Once the bootable volume is configured, you will take a snapshot of the volume and register the snapshot as a new EMI. Lastly, you will boot an EBS-backed instance from the new EMI.

Lab Objectives:

- Create an bootable Eucalyptus volume
- Snapshot a bootable volume
- Register a snapshot as an EMI
- Boot an EBS-backed instance
- Stop, start, reboot, and terminate an EBS-backed instance

Create a bootable Eucalyptus volume

In this section of the lab you will create a *helper* instance that will be used to help configure a bootable Eucalyptus volume. This lab requires the use of an instance with 10GB of disk space but only a single CPU. In order to create this instance, you will launch an instance specifying the use of the *c1.medium* vmtype. You will first create the instance and then copy a preconfigured image file containing a CentOS boot disk to the instance. You will then create a volume, attach the volume to the instance, and copy the image file to the volume.

1. Desktop From the Debian desktop, if necessary use SSH to log into the front-end host.

```
# ssh <front_end_public_IP>
```

2. Front End Create the *helper* instance that will be used to configure the bootable volume. Create the instance using the *c1.medium* vmtype. This vmtype has 10GB of total storage which results in the `/mnt` file system having 7GB of useable storage space when the instance boots.

```
# euca-describe-images
# euca-run-instances –k <keypair_name> -t c1.medium emi-<nnnnnnnn>
```

Note: The `euca-describe-availability-zones` verbose command displays information about vmtypes.

3. Front End Create a 2GB volume and once it is ready, attach it to your running instance as device `/dev/vdb`.

```
# euca-describe-availability-zones
# euca-create-volume –s 2 –z <cluster_name>
# euca-describe-volumes
# euca-attach-volume –i i-<nnnnnnnn> -d /dev/vdb vol-<nnnnnnnn>
# euca-describe-volumes
```

4. Desktop On the Debian desktop, open an xterm window.

5. Desktop From the Debian xterm window, use Secure Copy (`scp`) to copy the bootable CentOS image file `/root / bfebs-centos-5.8.img.gz` to the `/mnt` directory on the instance. Close the xterm window when finished.

```
# cd /root
# ls
# scp –i <key_file> bfebs-centos-5.8.img.gz <instance_public_IP>:/mnt
# exit
```

Note: If the `scp` command fails with the message `WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED` then you will need to edit the `/root/.ssh/known_hosts` file on the Debian desktop and remove the entry for the public IP address of the instance. Once done, reenter the `scp` command again.

6. Front End From the front-end host, use SSH to log in to the running instance.

```
# cd /root
# ls
# ssh –i <key_file> <instance_public_IP>
```

7. Front End On the instance, unzip the bootable image in `/mnt` . This is a large file and the command will take a few minutes to complete.

```
# cd /mnt
# ls
# gunzip bfebs-centos-5.8.img.gz
```

8. Front End On the instance, write the bootable image file to the volume attached to the instance. Before writing the file, first verify that the device `/dev/vdb` exists. This is a large file and this command will take several minutes to complete.

```
# ls /dev/vd*
# dd if=bfebs-centos-5.8.img of=/dev/vdb bs=10M
```

9. Front End Exit the SSH session to the instance.

```
# exit
```

10. Front End From the front-end host, detach the volume from the instance.

```
# euca-describe-volumes
# euca-detach-volume vol-<nnnnnnnn>
# euca-describe-volumes
```

11. Front End From the front-end host, delete the *helper* instance as it is no longer needed.

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

Snapshot a bootable volume

Front End

In this section of the lab you will create a snapshot of the volume with the bootable file system on it.

1. From the front-end host, create a snapshot of the bootable volume. This will take several minutes to complete.

```
# euca-describe-volumes
# euca-create-snapshot vol-<nnnnnnnn>
```

2. Monitor the progress of the snapshot creation. Wait until the output reports a state of `completed` before moving to the next step.

```
# euca-describe-snapshots
```

Note: Snapshot creation can take a few minutes.

3. Delete the volume as it is no longer needed.

```
# euca-delete-volume vol-<nnnnnnnn>
```

4. Verify that there are no volumes configured on the Storage Controller.

```
# euca-describe-volumes
```

Register a snapshot as an EMI

Front End

In this section of the lab you will register the snapshot of the bootable volume as a new EMI.

1. Register the snapshot as a new EMI image.

```
# euca-register –n bfebs-centos –s snap-<nnnnnnnn>
```

2. Display the new EMI.

```
# euca-describe-images
```

Boot an EBS-backed instance

Front End

In this section of the lab you will boot a new EBS-backed instance from the snapshot-backed EMI. As the instance is launched, a new bootable volume will automatically be created from the snapshot registered as an EMI. This volume will be attached to the instance, allowing the instance to boot.

1. From the front-end host, how many volumes are configured on the Storage Controller? There should not be any. If there are, delete them now.

```
# euca-describe-volumes
```

2. View the new snapshot-backed EMI.

```
# euca-describe-images
```

3. Launch an EBS-backed instance from the new EMI. In this case the instance has a valid root password so the `-k <key_name>` option is not necessary.

```
# euca-run-instances emi-<nnnnnnnn>
```

4. Monitor instance startup until it is listed with a status of `running` .

```
# euca-describe-instances
```

5. From the front-end host, use SSH to log in to the new instance. The root password is *foobar* .

```
# ssh <instance_public_IP>
```

Remain logged in until told to log out.

Stop, start, reboot, and terminate an EBS-backed instance

In this section of the lab you will test the persistence of data in an EBS-backed instance when it is stopped and restarted. You will do this by creating a file in the instance, stopping the instance, and then restarting it. You will also reboot the instance. Finally, you will terminate the instance.

1. While logged into the EBS-backed instance, make a change to the file system and then log out.

```
# touch /root/testfile
# exit
```

2. Stop the instance.

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

3. View the state of the EBS-backed instance.

```
# euca-describe-instances
```

What is the reported instance state? Wait until it reports a state of `stopped`.

4. Start the instance again. Note that the instance might not get the same private IP address again.

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

5. Use SSH to log in to the instance again using the password *foobar* . List the contents of the `/root` directory.

```
# ssh <instance_public_IP>
# ls /root
```

Is your `testfile` still there?

6. Log out of the instance.

```
# exit
```

7. Reboot the instance and immediately view the state of the instance.

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

   Does it quickly report `running` again? While the instance state is `running`, the software within the instance is still rebooting.

8. Use SSH to log into the instance using the password *foobar* . If SSH fails, wait a minute and try again.

```
# ssh <instance_public_IP>
```

9. List the contents of the `/root` directory.

```
# ls /root
```

   Is your `testfile` still there?

10. Exit the SSH session to the instance.

```
# exit
```

11. From the front-end host, terminate the instance.

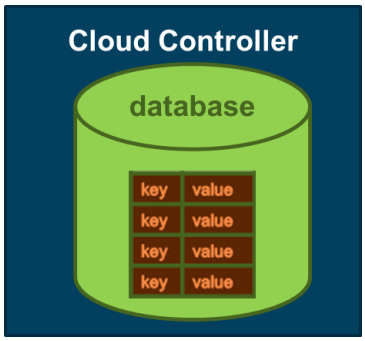
```
# euca-describe-instances
# euca-terminate-instances i-<nnnnnnnn>
```

12. If you were to launch another EBS-backed instance, would it have the file `/root/testfile`? Why or why not?

Eucalyptus Metadata Service

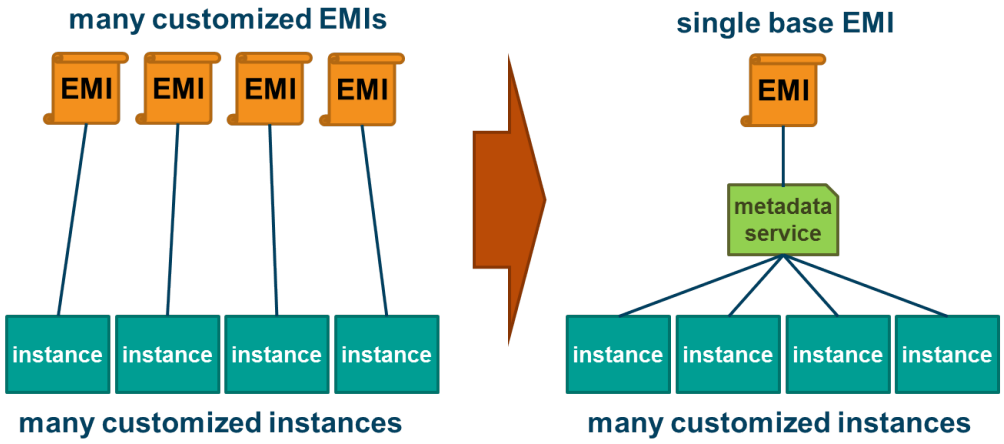
Eucalyptus provides an Amazon Web Service-compatible metadata service that provides running instances access to instance-specific information from the Cloud Controller's database. The metadata service is designed to provide small amounts of information to an instance when, or after, it is launched. The information provided can be standard pieces of information about the instance as defined by Eucalyptus or small pieces of information supplied by the user when launching the instance.

The Eucalyptus-defined data is often referred to as metadata. The user-defined data is often referred to as userdata. The data is stored as key/value pairs in the Cloud Controller's database.

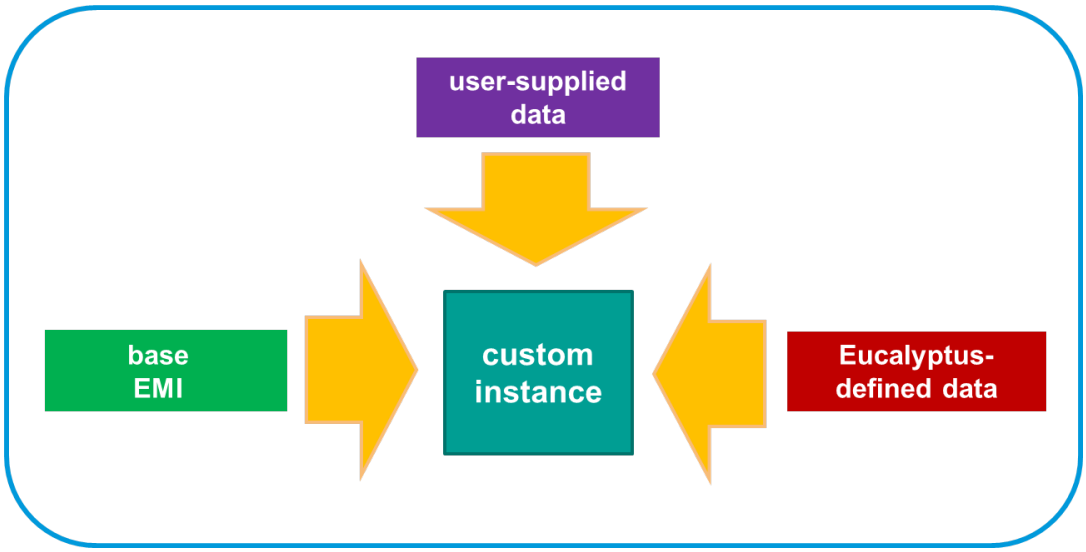


Metadata Service Benefits

The primary benefit of the metadata service is to reduce the need to have to create and manage multiple EMIs. For example, consider a situation where four instances with slightly different configurations are required. Without the metadata service, you would have to create four separate EMIs and use each to launch an instance. However, with the metadata service you might be able to create a single EMI and simply customize the instances that are booted from it. The customization typically occurs by using start-up scripts in the instance to download and process the metadata and userdata. This means that you would have to include the start-up scripts in the EMI when you create it so that they can be transferred to the instances when they launch.



This diagram below illustrates the three possible inputs that can be leveraged to customize an instance at start-up. The actual configuration of the instance is a combination of the information in the boot image along with Eucalyptus-defined (metadata) and user-supplied (userdata) information.



Metadata Service Access

While the metadata service is available in all four Eucalyptus network modes, access to it from the instances differs depending on the network mode. The metadata service is accessed by the instances using a URL.

- In either MANAGED or MANAGED-NOVLAN network mode, use the URL:
  - http://169.254.169.254/latest/<specific_metadata_information_request>

    Note: The Cluster Controller's maps the Automatic Private IP Address of 169.254.169.254 to the actual Cloud Controller IP address and port 8773.
- In SYSTEM of STATIC network mode, use the URL:
  - http://<CLC_IP>:8773/latest/<specific_metadata_information_request>

The `/latest` directory is always the root of metadata information in the Cloud Controller.

Eucalyptus recommends that you configure a DNS entry for the Cloud Controller that maps to the appropriate IP address based on your network mode and network address scheme. This way you can configure your instances to access the metadata service using a DNS name.

Metadata Keys

From an instance, the list of available metadata keys can be downloaded from the Cloud Controller using standard Linux command-line tools. For example:

```
curl -m 10 -s http://169.254.169.254/latest/meta-data > metadatakey.txt
```

The command above downloads all available metadata key names and writes them to a file called `metadatakey.txt`, which can then be either viewed or manipulated by other scripts.

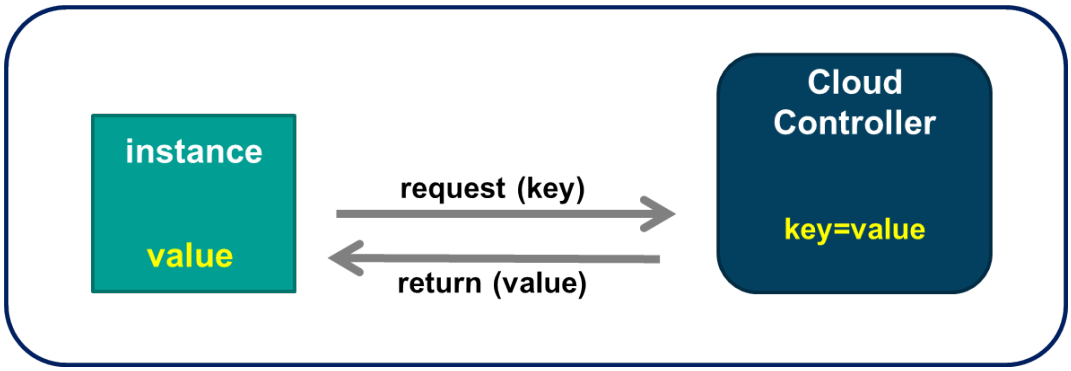
The following table lists the available metadata key names in Eucalyptus 3.2.

block-device-mapping	security-groups	ami-manifest-path
reservation-id	public-keys	public-keys/0
kernel-id	ramdisk-id	ami-launch-index
local-ipv4	instance-type	local-hostname
public-ipv4	hostname	public-hostname
ami-id	instance-id	

Fetch a Metadata Value

From an instance, you can use the command shown below to fetch the value associated with a specific metadata key name. In the case where there are nested key names below a key name, the names of the nested key name will be returned instead. Unless you know the last key name in a series of nested key names, you could use an iterative set of `curl` commands to walk down the list of key names in order to get their values.

```
curl -m 10 -s http://169.254.169.254/latest/meta-data/security-groups \
   > security-groups.txt
```



Metadata Example

The example below shows how to use a `curl` command in an instance's start-up script to download the user's public key into the instance's Secure Shell (SSH) `authorized_keys` file. Populating the `authorized_keys` file allows the user to use their private key to authenticate to the instance during an SSH log in without knowing, supplying, or even having a root password.

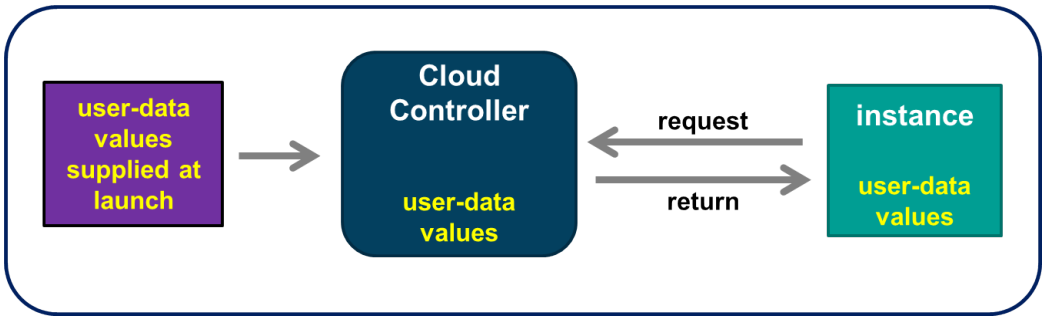
To accomplish this, add the following to the EMI's `/etc/rc.local` file:

```
mkdir –p /root/.ssh
#
touch /root/.ssh/authorized_keys
curl –retry 2  -retry-delay 5 –m 45 –s /
   http://169.254.169.254/latest/meta-data/public-keys/0/openssh-key /
   | grep 'ssh-rsa' >> /root/.ssh/authorized_keys
```

Note: The `curl` command shown above is actually a single entry in the `/etc/rc.local` file. The fact that it is shown as three lines in the illustration above is a result of the limited line width available on the page.

Userdata

A user can supply a small amount of data to the instance when they launch it. The data can be provided directly from command-line arguments or the Eucalyptus User Console, or indirectly from a file specified from the command line or the Eucalyptus User Console. What you provide to the database is what you get back from the database. In order to make use of this data, the instance's start-up scripts must be able to parse this userdata.



Creating Userdata

Userdata can be created using euca2ools with the following command:

```
euca-run-instances -k <keyname> --user-data /
   <user_supplied_text> emi-<nnnnnnnn>
```

The `euca-run-instances` command also offers a `--user-data-file <file_name>` option that allows the user to place the userdata in a file.

The Eucalyptus User Console also allows userdata to be supplied using a field in the **Launch new instance** widow:

Whatever userdata is supplied to the instance at launch is stored in the Cloud Controller database. It is not retrieved until a user or start-up script in the instance requests it.

Userdata Example

This example will use a combination of metadata and userdata, along with a start-up script, to have an instance automatically attach a volume at boot. The start-up script will also partition the volume, create a file system on the partition, and mount the file system to a directory named `/ebs`.

The start-up script assumes that the volume has been created ahead of time and that the user will supply the volume ID as userdata when they launch the instance. The start-up script also assumes that the euca2ools commands have been installed in the EMI and therefore are available in the instance when it launches.

Because the start-up script runs a euca2ools command to attach the volume, the instance must be able to authenticate to Cloud Controller. For this reason the user must also supply the standard EC2 authentication information as userdata when the instance is launched. The necessary authentication information can be copy and pasted from the user's `eucarc` file.

Notice that all the userdata shown below is in the form of `export` commands that set environment variables. These commands will be written to a file on the instance and then that file will be read by the operating system's shell program in order to set the environment variables. The first three environment variables will be automatically used by the `euca-attach-volume` command to authenticate to the Cloud Controller. The last environment variable supplies the volume ID number to the `euca-attach-volume` command.

Launch an instance with the following user data:

```
export EC2_URL=<url>
export EC2_ACCESS_KEY=<key>
export EC2_SECRET_KEY=<key>
export EC2_VOL_ID=<vol-id>
```

Note: The instance would have to be launched with the actual values of the environment variables and not the angle-bracket values as shown above. The actual EC2 URL and KEY values can be copied and pasted from the user's `eucarc` file. The volume ID value would come from the output of the `euca-describe-volumes` command.

Add the following to the EMI's `/etc/rc.local` file:

```
INSTANCE_ID=$(curl --retry 2 --retry-delay 5 -m 45 -s /
   http://169.254.169.254/latest/meta-data/instance-id)
#
curl --retry 2 --retry-delay 5 -m 45 -s /
   http://169.254.169.254/latest/user-data/ >> /
   /root/.env_variables
#
source /root/.env_variables
#
euca-attach-volume -i $INSTANCE_ID -d /dev/vdc $EC2_VOL_ID
#
sleep 5
#
fdisk /dev/vdc < /partition_script
#
mkfs.ext3 /dev/vdc1
#
mkdir -p /ebs
#
mount /dev/vdc1 /ebs
```

Note:

The `/partition_script` would need to exist in the EMI in order to be available in the running instance. It must contain the following six lines:

n

p

1

(press the Enter key for a blank line)

(press the Enter key for a blank line)

w

It would create a (*n*)ew partition, it would be a (*p*)rimary partition, it would be partition (*1*), it would use a default starting cylinder, it would use a default end cylinder, and would (*w*)rite the partition table to the disk and exit.

The script to attach a volume at boot was purposely kept relatively simple for training purposes. In the real world, the script could be enhanced, for example, to include commands that would create the volume, find it's volume ID, and then attach the volume.

Optional Lab - Working with Metadata Services

In this optional lab exercise you will work with Eucalyptus-defined instance metadata and user-supplied userdata, both of which are accessible through the Eucalyptus metadata services. From an instance, you will download from the Cloud Controller the list of available metadata keys and then view the values associated with a few keys. Then you will customize an EMI image so that when it is used to launch an instance, that instance will work with user-supplied userdata to automatically attach to a volume, partition the volume, create a file system on the volume, and then mount the file system.

Lab Objectives:

- View Eucalyptus-defined metadata keys and values
- Modify an EMI to use metadata services
- Launch an image with user-supplied userdata

View metadata keys and values

In this section of the lab you will create an instance. After logging in to that instance you will download the list of available Eucalyptus-defined metadata keys and then view the values associated with a few of the keys.

1. Desktop From the Debian desktop, if necessary use SSH to log in to the front-end host.

```
# ssh <front_end_public_IP>
```

2. Front End Launch an instance using the CentOS EMI in the *kvm-centos* bucket.

```
# euca-describe-images
# euca-run-instances –k <keypair_name> emi-<nnnnnnnn>
```

3. Front End Once the instance is running, use SSH to log in to the instance.

```
# cd /root
# ls  # ssh –i <key_name> <instance_public_IP>
```

4. Front End From the instance, use the `curl` command to download the list of available predefined metadata keys into the text file `metadatakeys.txt`.

```
# curl -m 10 -s http://169.254.169.254/latest/meta-data > metadatakeys.txt
# ls
```

5. Front End From the instance, display the contents of the `metadatakeys.txt` file.

```
# cat metadatakeys.txt
```

Note: Some of the key names are appended with a forward-slash character. This indicates that those keys are actually the top of a hierarchy of keys and values. You will see how to navigate down these hierarchies later in the lab.

6. Front End From the instance, view the value associated with the `security-groups` key. Which security group was the instance launched in?

```
# curl -m 10 -s http://169.254.169.254/latest/meta-data/security-groups
```

Note: The value associated with the `security-groups` key actually prints on the screen *before* the bash shell prompt. Press the Enter key to get a normal shell prompt again.

7. Front End From the instance, traverse down through the keys and values hierarchy associated with the `public-keys` key.

```
# curl -m 10 -s http://169.254.169.254/latest/meta-data/public-keys
# curl -m 10 -s http://169.254.169.254/latest/meta-data/public-keys/0
# curl -m 10 -s http://169.254.169.254/latest/meta-data/public-keys/0/
openssh-key
```

Note: The `value 0=adminkey` which is displayed by the second command, denotes that key `0` is the key associated with the key pair labeled `adminkey`.

8. Front End Exit out of the instance.

```
# exit
```

9. Front End Terminate the instance.

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

Modify an EMI to use metadata services

Front End

In this section of the lab you will modify an existing EMI so that it will use metadata services to customize the instance as it boots. Specifically, you will modify the `/etc/rc.local` file in the EMI to use Eucalyptus-defined metadata and user-supplied userdata to attach to and configure a volume at boot time.

1. From the front-end host, change directory to `/var/images`. List the contents of the directory, which should list the image file `kvm-centos-5.8.img`. This is the image file that was modified in an earlier lab to include NTP and a repaired `/etc/fstab`.

```
# cd /var/images
# ls
```

2. Use a loopback mount to mount the image file to the `/var/tempmnt` directory and verify that it was mounted.

```
# mount –o loop kvm-centos-5.8.img /var/tempmnt
# mount
```

3. Use an editor (either `vi` or `nano –w`) to create the file `/var/tempmnt/partition_script`. This script will be called as an argument to the `fdisk` command when the instance boots. The contents of the script will create a partition on a volume. When creating the contents, enter only the characters and blank line shown below, do not add any other lines to this file.

```
# vi /var/tempmnt/partition_script
n
p
1
1
(just press Enter for a blank line)
w
```

Note: The `partition_script` file will be located in the `/` directory of the instances when they boot from this EMI.

4. Change to the `etc` directory in the mounted image file. Once there, list the contents of the directory and note the `rc.local` file.

```
# cd /var/tempmnt/etc
# ls
```

5. Edit the `rc.local` file (using either `vi` or `nano –w`) and add the following lines to the end of the file. Note that in these lab instructions, the entries starting with `INSTANCE_ID` and `curl` line wrap but should be entered as continuous lines in the actual `rc.local` file.

```
# vi rc.local

INSTANCE_ID=$(curl --retry 2 --retry-delay 5 -m 45 -s
 http://169.254.169.254/latest/meta-data/instance-id)

curl --retry 2 --retry-delay 5 -m 45 -s http://169.254.169.254/latest/
user-data/ >> /root/.env_variables

source /root/.env_variables

ntpdate pool.ntp.org

/usr/bin/euca-attach-volume -i $INSTANCE_ID -d /dev/vdb $EC2_VOL_ID

sleep 20

fdisk /dev/vdb < /partition_script
mkfs.ext3 /dev/vdb1
mkdir -p /ebs
mount /dev/vdb1 /ebs
touch /ebs/here_is_your_volume
```

6. Unmount the image file and verify that it was unmounted.

```
# cd
# umount /var/tempmnt
# mount
```

7. Bundle, upload, and register the image as a new EMI. Upload the image to a Walrus bucket named *meta-centos*.

```
# cd /var/images
# euca-bundle-image –i kvm-centos-5.8.img
# euca-upload-bundle -b meta-centos -m /tmp/kvm-
centos-5.8.img.manifest.xml
# euca-register meta-centos/kvm-centos-5.8.img.manifest.xml
```

Launch an instance with user-defined data

Front End

In this section of the lab you will create a volume as well as create a file with several lines of userdata. Then you will use euca2ools to launch an instance, adding the necessary command arguments to read the userdata file. Finally, you will verify that the volume is attached to the instance, partitioned, has a file system, and the file system is mounted.

1. On the front-end host, create a new 2GB volume. Wait for it to become available.

```
# euca-create-volume -s 2 -z <cluster_name>
# euca-describe-volumes
```

Note: You can use the `euca-describe-availability-zones` command to see your cluster name.

2. Look at your screen and write down the volume ID number. You will need the ID number for a future lab step.

3. Desktop On your Debian desktop, open another xterm window.

4. In the new xterm window, use SSH to log in to the front-end host. You should have now be logged in to the front-end host from two different xterm windows.

```
# ssh <front_end_public_IP>
```

5. Front End In the new xterm window while logged in to the front-end host, display the contents of the `/root/.euca/eucarc` file.

```
# cat /root/.euca/eucarc
```

6. In the original xterm window that is logged in to the front-end host, use an editor (either `vi` or `nano -w`) to create a new file name `mydata` in the `/root` directory of the front-end host. Add the four lines of userdata shown below to the file. Copy the keys the `eucarc` file and paste them to the `mydata` file. IMPORTANT! You do not need the single quotes that appear in the `eucarc` file. Close the file when finished.

```
# vi /root/mydata

export EC2_URL=http://<front_end_public_IP>:8773/services/Eucalyptus
export EC2_ACCESS_KEY=<key_value_from_the_eucarc_file>
export EC2_SECRET_KEY=<key_value_from_the_eucarc_file>
export EC2_VOL_ID=<enter_the_volume_ID_from_the_earlier_lab_step>
```

7. Close the second open xterm window (the one where you ran the `cat /root/.euca/eucarc` command.

```
# exit
```

8. In the remaining xterm window that is logged in to the front-end host, launch a new instance using the `mydata` file and the EMI in the *meta-centos* Walrus bucket.

```
# cd /root
# euca-describe-images
# euca-run-instances -k <keypair_name> -f mydata emi-<nnnnnnnn>
```

9. Use SSH to log in to the instance once it is running.

```
# ssh –i <key_file> <public_IP_of_instance>
```

10. In the SSH session to your instance, verify that the volume is partitioned and mounted to the directory `/ebs`.

```
# fdisk –l /dev/vdb
# mount
# ls –l /ebs
```

11. Exit the SSH session to the instance.

```
# exit
```

12. Terminate the instance.

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

Eucalyptus Identity and Access Management Introduction

Eucalyptus Identity and Access Management (EIAM) is a large subject and as such, this lesson will cover a number of topics including:

- Accounts, users, groups, and resources
- LDAP integration
- Security credentials
- Eucalyptus resource names
- Access control policies
- Quotas

In addition, this lesson will cover the process of creating, viewing, and deleting accounts, groups, and users from both the Eucalyptus Administrator Console and command line.

Eucalyptus IAM

Eucalyptus Identity and Access Management (EIAM) is an authentication, authorization, and accounting system which manages user identities, enforces access controls over resources, and provides the basis for reporting on cloud resource usage.

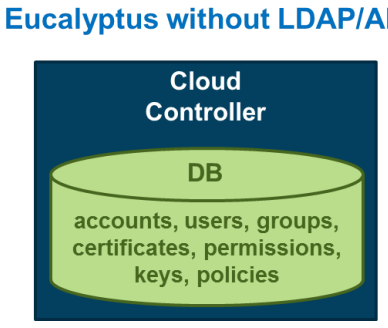
EIAM is based on the AWS IAM organizational model and authorization scheme. It also provides two extensions that are appropriate for a private cloud environment. These extensions are support for quotas and more granular access control of resources.

Quotas are important in a private cloud because, unlike a public cloud, they often have more limited capacity that has to be shared among the cloud users. Quotas help prevent a single user or even a group of users from consuming all the cloud resources. More granular access control allows the cloud or account administrators to control which resources can be used by which users or groups of users.

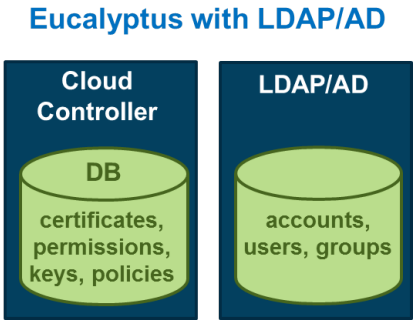
LDAP and AD Integration

EIAM includes support for optional integration with either LDAP or Active Directory.

By default, all cloud account, user, and group configuration information is stored in the Cloud Controller database as shown below.



However, to simplify integration with other datacenter applications and management tools, cloud account, user, and group configuration information can be moved to LDAP or Active Directory. Credentials, keys, permissions, and policy information remain stored in the Eucalyptus database.



If you integrate Eucalyptus with LDAP or AD, only normal users are imported from LDAP/AD. At this point Eucalyptus management of accounts, normal users, and groups is no longer supported. The *admin* account users remain in the Cloud Controller database and are managed by Eucalyptus. Currently there is no way to map individual users from LDAP to Eucalyptus accounts. You can only map LDAP groups (with users) to Eucalyptus accounts.

The primate step in LDAP integration is to create an LDAP Integration Configuration (LIC) file using `/usr/sbin/euca-lictool --password <LDAP_password> --out example.lic`. Once the file has been created, you would edit the file and add your specific LDAP information. Once this file edits are complete you would then run `/usr/sbin/euca-modify-property -f authentication.ldap_integration_configuration=<lic_filename.lic>` to install and activate LDAP. An example LIC file can be found at `/usr/share/eucalyptus/lic_template`.

For more detailed instruction about LDAP integration, see the *Eucalyptus Administration Guide* at *http://www.eucalyptus.com/docs*.

EIAM Accounts

An account include users, groups, and either all or some subset of cloud resources.

Accounts should be created that represent the divisions or departments within the company or organization. For example, if a company has both engineering and sales departments, then two accounts could created to represent them. For example, members of the engineering department could be added as users to the *engineering* account while members of the sales department could be added as users to the *sales* account.

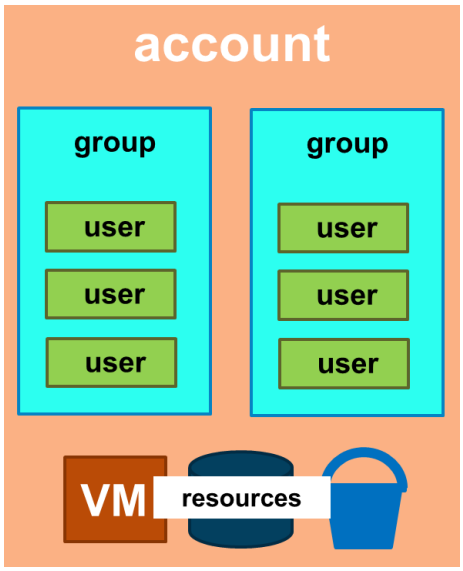
Accounts are the primary unit for resource usage accounting. An account centrally controls all of the resources under its span of control. This includes the account itself, as well as users, groups, and additional resources.

Accounts are also a separate namespace for users, groups, security groups, and key pairs. For example, a user named Janet can exist in two separate accounts.

Note: The same is true for group names, security groups, and key pairs. As long as they are in different accounts, duplicate names can exist. Within an account, all objects must have unique names.

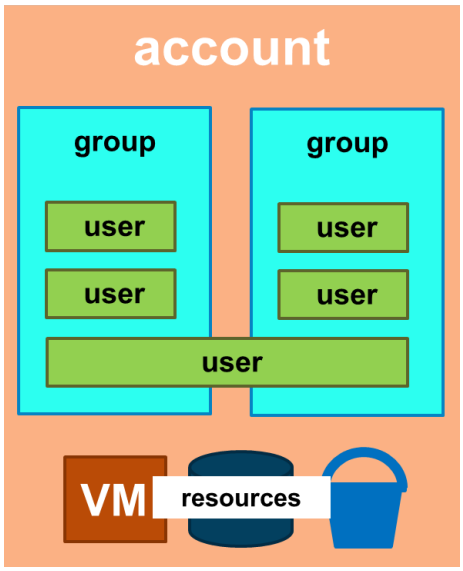
Groups are used to assign resource access controls to a set of users. This simplifies both the initial cloud configuration as well as an subsequent cloud configuration changes.

Note: Any permissions created for users or groups within the account do not apply to the account itself.



Account Users

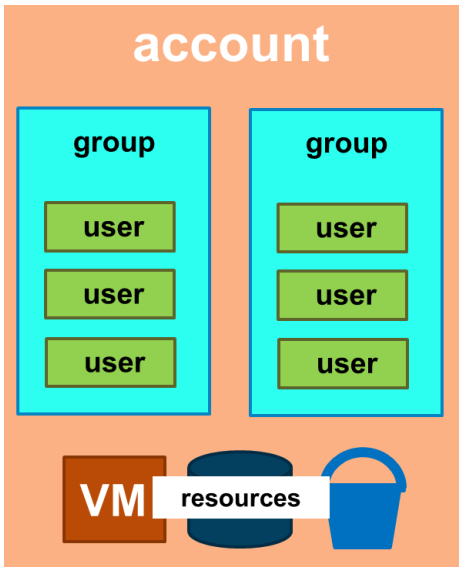
Accounts include users. A user is an individual person who interacts with the cloud and has a unique name within the account. Users are associated with a single account, thus users with the same name can be configured as long as they are in different accounts. Users can belong to multiple groups within the same account. User names are case insensitive.



Accounts and Resources

Accounts also include resources, which are entities or objects with which users interact. These can include other instances, volumes, buckets, images, IP addresses, availability zones, key pairs, snapshots, and vmtypes.

Resources have friendly names as well as Eucalyptus resource names (ERNs). Friendly names can generally be used as command-line arguments, whereas ERNs must be used when creating access control policies.



Special Identities

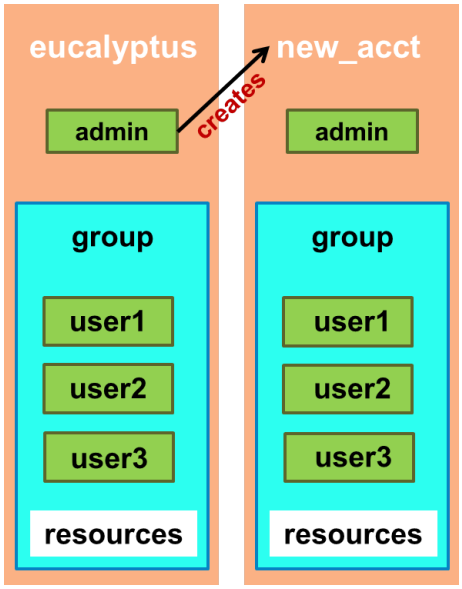
The *eucalyptus* account and its *admin* user are created during cloud software installation. The *admin* user in this account is a cloud administrator and has full privileges in the cloud. Any other user created in the *eucalyptus* account is also a cloud administrator with full privileges. Users in the *eucalyptus* account are also sometimes referred to as system administrators.

Note: You cannot remove the *eucalyptus* account from the cloud.

By default, only the *admin* user in the *eucalyptus* account can create new accounts. When you create a new account, a new *admin* user is also created for that account. The *admin* user has access to all resources assigned to that account by the *eucalyptus* account administrator. The *admin* users in these other accounts are referred to as an account administrators.

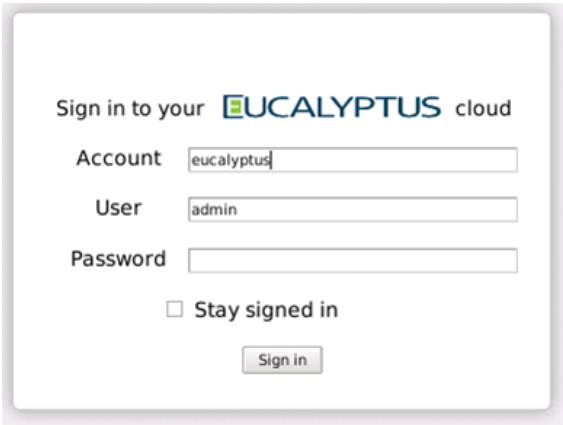
The cloud or account administrator can delegate resource access to other users or groups in the account by using access control policies.

The *admin* user in any account cannot be removed.



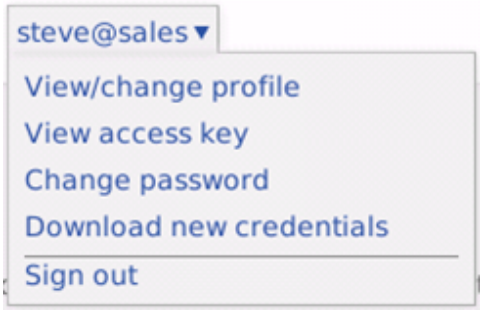
Login Profile

A login profile consists of a login name and password. A login profile is required for any user to access the Eucalyptus Administrator Console. Cloud or account administrators will need to create a user name and password for any user before they will be able to log in to the Administrator Console. While the Administrator Console is primarily a cloud administration tool, it can also be used by normal users to change their own password and download their own cloud credentials (keys, certificate, and `eucarc` file).



Downloading Credentials

By default, a new cloud user does not have credentials. Users can create and download their own credentials once they log into the Eucalyptus Administrator Console. However, even a user with credentials they will still need to be given permission to use cloud resources. These permissions are granted by a cloud or account administrator when they create access control policies.



Eucalyptus Resource Names

Eucalyptus Resource Names (ERNs) are the name format used to uniquely identify cloud resources, users, and groups. They look like this:

arn:aws:<vendor>:<region>:<namespace>:<relative-id>

- *arn*: Amazon Resource Name - used in ERN for AWS compatibility
- *aws*: Amazon Web Services compatible
- *vendor*: identifies the service (for example, ec2, s3, or iam)
- *region*: specifies the region of a resource (not used in Eucalyptus)
- *namespace*: specifies the name space of a resource
- *relative-id*: specifies the ID within the service and the namespace

ERNs are seen in Eucalyptus Administrator Console and command-line output, and are used when you write EIAM access control policies. For this reason it is important that you are able to recognize them, interpret them, and even write them.

IAM and S3 ERN Examples

The pattern for an IAM ERN looks like this:

arn:*aws*:*iam*::<*account id*>:[*user*|*group*]/<*optional path*>/<*user or group name*>

Note: Note the double **::** between *iam* and <*account id*>. This blank field is the *region* field and is not used in Eucalyptus. It is only used in Amazon Web Services.

Examples:

- arn:aws:iam::eucalyptus:user/steve
- arn:aws:iam::eng:user/*
- arn:aws:iam::sales:group/west/salesteam

The first example above describes a user named *steve* who is a member of the *eucalyptus* account. The second example describes all users in the *eng* account. The asterisks is known as a wildcard character. The last example describes a group named *salesteam* that is in the *sales* account. The *salesteam* group was create with an optional path of */west*. Optional paths, and the reasons for, them are covered later in this lesson.

Note: The Eucalyptus Administrator Console prints account names. The command-line tools only display the account ID numbers.

The pattern for an S3 ERN looks like this:

arn:*aws*:*s3*:::<*bucket name*>[/<*key name*>]

Note: Note the triple **:::** between *s3* and <*bucket name*>. S3 ERNs do not specify a *region* or an *account_name*.

Examples:

- arn:aws:s3:::acme_bucket/emi-1234abcd
- arn:aws:s3:::acme_bucket/*

The first example describes a specific EMI that resides in a Walrus bucket named *acme_bucket*. The second example describes all data objects that reside in a Walrus bucket named *acme_bucket*.

EC2 ERN Examples

The format of Eucalyptus EC2 ERNs is different from AWS EC2 ERNs. The Eucalyptus EC2 ERN format includes extensions that provide more granularity than AWS. This granularity is useful to provide more fine-grained resource control in access control policies. AWS only supports the wild card "*" character in AWS IAM access control policy statements. This means that in Amazon IAM you cannot constrain users to specific resources. Eucalyptus IAM, on the other hand, extends resource ERNs to include specific resources.

The pattern for an EC2 ERN looks like this:

arn:*aws*:*ec2*::<*account_ID*>:<*resource_type*>/<*resource_ID*>

Note: Note the double **::** between *ec2* and <*account_ID*>. This is the blank *region* field because the Eucalyptus private cloud does not use regions like the AWS public cloud.

Note: An account ID is optional, but can be used to limit resource access to particular accounts. If an Account ID is not provided, the pattern would have a triple **:::** between *ec2* and <*resource_type*>.

Examples:

- arn:aws:ec2:::vmtype/m1.small
- arn:aws:ec2::marketing:image/emi-af45e531

The first example describes the vmtype named *m1.small*. The second example describes a specific EMI. Notice also that the second example refers to a specific account as well, the *marketing* account. Including a specific account name in an EC2 ERN might be useful if this ERN were included in an access control policy statement. It could be used to limit access to the resource to users in a specific account.

In these two examples, two types of EC2 resources were described, a *vmtype* resource and an *image* resource. Other resource types may also be described. Eucalyptus supports the following EC2 resource types:

- image
- securitygroup
- address
- availabilityzone
- instance
- keypair
- volume
- snapshot
- vmtype

Note: For the address resource type, either a single IP address or an IP address range (for example: 192.168.7.1-192.168.7.254) is supported.

Access Control Policy Overview

By default, a user has very limited permissions in the cloud. They can only:

- Download their credentials
- List information about themselves
- List the account they are in

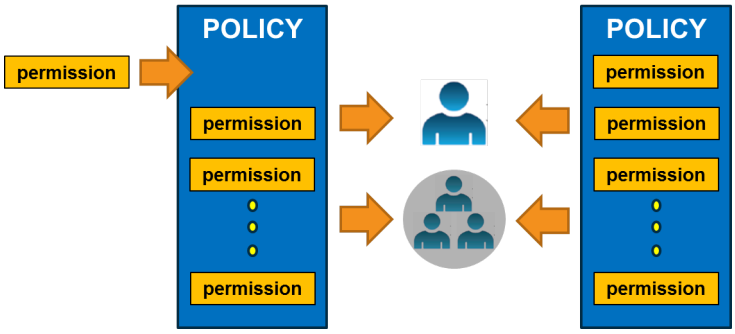
A cloud or account administrator can create an access control policy. This policy is a document that provides a formal statement of one or more permissions allowing or disallowing access to resources.

Policies can be assigned to a user or a group. They are assigned by a cloud or account administrator. The user or group receives the permissions in the policy. Multiple policies can be assigned to the same user or group.

Policies, Users, and Groups

A permission can be added to a policy. An access control policy can be assigned to a user or group. In fact, two or more access control policies can be assigned to a user or group.

Note: Accounts should not be assigned access control policies. The results of doing so has not been tested by Eucalyptus.



Policy Language

Policies are written in JavaScript Object Notation (JSON) format, which is a text-based data interchange format compatible with the AWS IAM policy language. A policy can contain one or more permission statements, which specify whether to *Allow* or *Deny* a list of actions to be performed on a list of resources under certain conditions. Here is an example:

```
{
  "Version":"2008-10-17",
  "Statement":[{
    "Sid":"1",
    "Effect":"Allow",
    "Action":"*",
    "Resource":"*"
  }]
}
```

The *Version* is the access policy language version. This is an optional element. It often contains the date the policy was written.

The *Statement* is the main element for a policy. The *Statement* element contains an array of individual elements. Each individual element is a distinct JSON block enclosed in curly brackets { } and separated by a comma. For example; "Statement":[{...},{...},{...}].

The *Sid* is the optional statement ID. If statement IDs are used, each statement in a policy document should have a unique *Sid*. The *Sid* can include an alpha-numeric string.

The policy statement above, if applied to a user or group, would allow all actions on any resource (note the use of wild cards in the *Action* and *Resource* elements.) Such a user or group would have administration-like privileges in the cloud.

The *Statement* above does not include a *Condition* element. A later example will include a *Condition* element.

Policy Statements

While Eucalyptus has Eucalyptus Resource Names (ERNs), for compatibility with Amazon policy format, Eucalyptus uses *arn* in entries within policy statements. The table below defines sections of a policy document.

Component	Meaning
`Sid`	Optional statement ID, must be unique within the policy
`Effect`	Decision that applies to the resource; either `Allow` or `Deny`
`Action` or `NotAction`	User-specific commands on the resource, for example `ec2:RunInstances`. There are `ec2:*`, `s3:*`, and `iam:*` actions.
`Resource` or `NotResource`	Resource effected, specified as an ARN, for example `arn:aws:s3:::user1_bucket/centos.img`. There are `ec2:*`, `s3:*`, and `iam:*` resources.
`Condition`	Additional constraints on the permission, for example `DateGreaterThan` or `ec2:ExpirationTime`

The *NotAction* element is useful if you want to make an exception to a list of actions. The following example refers to all actions other than the EC2 RunInstances action.

"NotAction":"ec2":"RunInstances".

The *NotResource* element is useful if you want to make an exception to a list of resources.

Three categories of actions can be specified in an access control policy. EC2 actions (ec2:*) allow users to work with instances, images, volumes, snapshots, and IP addresses. S3 actions (s3:*) allows a user to work with buckets and data object within buckets. IAM actions (iam:*) allow a user to work with accounts, users, groups, credentials, and access control policies.

For more information about IAM actions, see *AWS Identity and Access Management API Reference* at *http://awsdocs.s3.amazonaws.com/IAM/latest/iam-api.pdf*.

For a list of S3 actions, see the section about the *S3 SOAP API Reference* at *http://docs.amazonwebservices.com/AmazonS3/latest/API/APISoap.html*.

For a list of EC2 actions see Actions at *http://docs.amazonwebservices.com/AWSEC2/latest/APIReference/Welcome.html?r=400*.

The *Condition* element is the most complex part of the policy statement. It is referred to as a condition block because although it has a single *Condition* element, it can contain multiple conditions, and each condition can contain multiple key-value pairs. When creating a condition block, you specify the name of each condition, and at least one key-value pair for each condition. An example of a condition is included later in the lesson.

Sample Policies

The following policy specifies permission to launch instances only if they are launched using the *m1.small* vmtype:

```
{
  "Version":"2008-10-17",
  "Statement":[{
     "Sid":"2",
     "Effect":"Allow",
     "Action":"ec2:RunInstances",
     "Resource":[
                  "arn:aws:ec2:::vmtype/m1.small“,
                  "arn:aws:ec2:::image/*",
                  "arn:aws:ec2:::securitygroup/*",
                  "arn:aws:ec2:::keypair/*",
                  "arn:aws:ec2:::availabilityzone/*",
                ]
              }]
}
```

Note: Note the effect if the resource was changed to the following:

"Resource":"arn:aws:ec2:::vmtype/*"

The next example restricts an instance's running time to 24 hours using an additional condition. 1440 is 1440 minutes, or 24 hours:

```
{
  "Version":"2008-10-17",
  "Statement":[{
     "Sid":"3",
     "Effect":"Allow",
     "Action":"ec2:RunInstances",
     "Resource":"*",
     "Condition":{
         “NumericEquals”:{
            “ec2:KeepAlive”:”1440”
         }
      }
  }]
}
```

The *ec2:KeepAlive* and *ec2:ExpirationTime* are Eucalyptus IAM extensions that are no present in AWS. If multiple values apply to a user, the longer value is used. For examples of each, see the *Administration Guide* at *http://www.eucalyptus.com/docs*.

This next example allows a user to run instances, manage security groups, volumes, snapshots, IP addresses, key pairs, as well as list and view these objects:

```
{
  "Statement": [{
        "Sid": "Stmt1313605116084",
        "Action": [
            "ec2:AllocateAddress",
            "ec2:AssociateAddress",
            "ec2:AttachVolume",
            "ec2:Authorize*",
            "ec2:CreateKeyPair",
            "ec2:CreateSecurityGroup",
            "ec2:CreateSnapshot",
            "ec2:CreateVolume",
            "ec2:DeleteKeyPair",
            "ec2:DeleteSecurityGroup",
            "ec2:DeleteSnapshot",
            "ec2:DeleteVolume",
            "ec2:Describe*",
            "ec2:DetachVolume",
            "ec2:DisassociateAddress",
            "ec2:GetConsoleOutput",
            "ec2:RunInstances",
            "ec2:TerminateInstances"
            "ec2:ReleaseAddress"
          ],
          "Effect": "Allow",
          "Resource": "*"
      } ]
}
```

For more detail about access control policies, and more examples of access control policies, see the *Administration Guid*e at *http://www.eucalyptus.com/docs* or *AWS Identity and AccessManagement* at *http://awsdocs.s3.amazonaws.com/IAM/latest/iam-ug.pdf*.

Paths and Policy Scope

An optional path is useful to identify organizational differences between users or groups in the same account. For example, consider the following two IAM ERNs:

- arn:aws:iam::sales:user/commercial/steve
- arn:aws:iam::sales:user/government/janet

Both the user *steve* and the user *janet* are members of the sales department and *sales* account. However, they sell to different customer segments. This can be easily noted on the screen by placing them in different paths where the path names indicate which customer segment they sell to.

More importantly, paths are used to control the scope of permissions or policies. An administrator could be given permission to manage only the users and groups in a very specific subdivision of the *sales* account (department). In the following example, this policy could be used to allow a specific administrator the privilege to only manage users and groups created in the sales account with *commercial* in their paths.

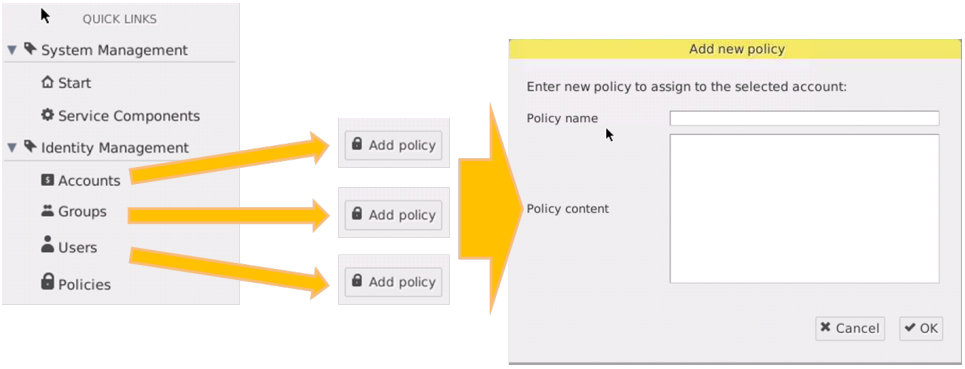
```
{
   "Statement":[{
      "Effect":"Allow",
      "Action":"iam:*",
      "Resource":["arn:aws:iam::sales:group/commercial/*",
                  "arn:aws:iam::sales:user/commercial/*"]
   }]
}
```

Note: Creating a user in a path does not determine group membership. Only explicit assignment to a group associates a user with a group.

It is recommended that you use separate accounts for resource accounting and chargeback, and that you use separate paths to limit permissions on resources.

Create Policies - Administrator Console

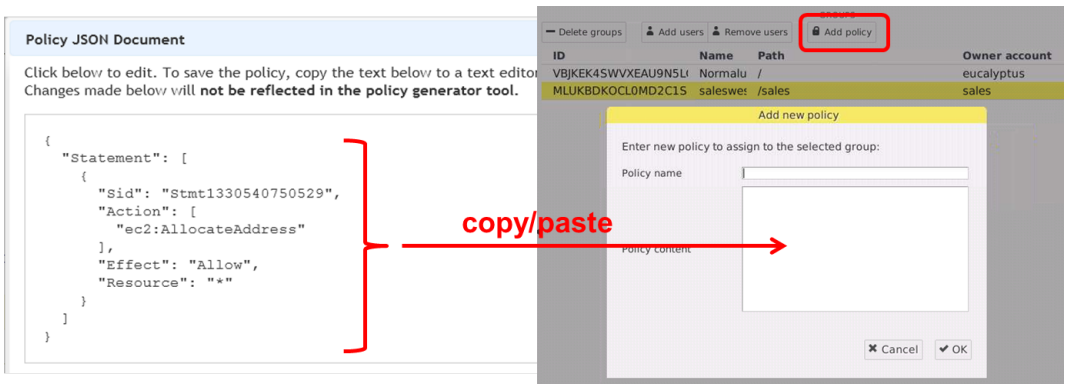
To create a new policy using the Eucalyptus Administrator Console, you would select either **Accounts**, **Groups**, or **Users** in the QUICK LINKS panel. On the main panel you would then select an account, group, or user and then click the **Add policy** button.



The Add new policy window will open and provide text boxes to enter the policy name and enter the policy itself using JSON-formatted text.

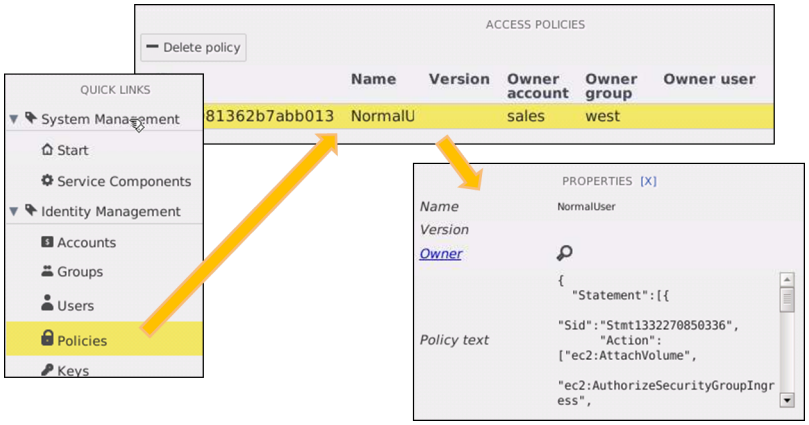
Amazon Policy Generator

It is possible to create a Eucalyptus-compatible policy using the online, browser-based Amazon Policy Generator. The JSON-formatted policy generated by the Amazon tool can then be copied and pasted into the Eucalyptus Administrator Console. Typically the policy generated can be used in Eucalyptus with only minor alterations.



View Policies - Administrator Console

You can view policies from the **Policies** link in the QUICK LINKS panel. Once you select **Policies**, a list of the cloud's current policies appears in the main window. Select a policy and its content appears to the right-side of the main panel. Notice that you can also delete a policy using the **Delete policy** button once you select a policy from the list in the main panel.



Manage Policies - Command Line

Starting with Eucalyptus 3, a new set of command-line commands were added. These commands all use the prefix `euare-*` (pronounced "you are"). The `euare-*` commands are used to manage EIAM functionality, including policies, accounts, users, and groups.

Group Policies

The `euare-groupaddpolicy` command builds a policy statement and adds it to a group. It has the following syntax:

```
euare-groupaddpolicy -g <group_name>  \
     -p <policy_name> -e [Allow|Deny]  \
     –a <actions> -r <resources> -o
```

It can only be run by a user with administrative privilege on the account - there is no `--delegate` option for this command.) For example, the command:

```
euare-groupaddpolicy -g west -e Allow -a "*" -r "*" -p allowall –o
```

The resulting policy would allow members of the group to perform any action on any resource.

Note: The quotes around the wild card characters are required.

Policies can also be uploaded from an external file using the following command:

```
euare-groupuploadpolicy –g group_name –p policy_name –f policy_file.txt \
     --delegate=account_name
```

The `euare-grouplistpolicies` command lists the names of policies associated with a group using the following syntax:

```
euare-grouplistpolicies –g <group_name> \
     <--delegate=account_name>
```

This command can be run by the administrator of the account, or if using the `--delegate` option, by a user in the eucalyptus account (a cloud administrator). If run by a cloud administrator, it would display the polices associated with the group in the account listed in the `--delegate` option.

Note: The content of the policies would also be displayed if the `–v` option was included.

There are additional options to limit the number of policies displayed on the screen and to paginate the display results. For more information, see the online help page for the command.

The `euare-groupgetpolicy` command displays the contents of a single named policy associated with the named group, but has no pagination options. It uses the following syntax:

```
euare-groupgetpolicy –g <group_name>   \
     -p <policy_name> <--delegate=account_name>
```

This command can be run by the administrator of the account, or if using the `--delegate` option, by a user in the eucalyptus account (a cloud administrator). If run by a cloud administrator, it would display the polices associated with the group in the account listed in the `--delegate` option.

The `euare-groupdelpolicy` command deletes the named policy from the named group. It has the following syntax:

```
euare-groupdelpolicy –g <group_name>     \
     -p <policy_name> <--delegate=account_name>
```

This command can be run by the administrator of the account, or if using the `--delegate` option, by a user in the eucalyptus account (a cloud administrator). If run by a cloud administrator, it would delete the polices associated with the group in the account listed in the `--delegate` option.

User Policies

The `euare-useraddpolicy` command builds a policy statement and adds it to a user. It has the following syntax:

```
euare-useraddpolicy –u <user_name> -p <policy_name> \
     -e [Allow|Deny|Limit] –a <actions>  -r <resources> -o
```

It can only be run by a user with administrative privilege on the account. (There is no `--delegate` option for this command.) For example, the following command:

```
euare-groupaddpolicy –u steve -e Allow -a "*" -r "*" -p allowall –o
```

...would allow the user steve to perform any action on any resource. The quotes around the wild card characters are required.

Note: Notice that there is also a `Limit` argument for the `–e` option. A `Limit` argument is used in quota policies rather than permission policies.

Policies can also be uploaded from an external file using the following command:

```
euare-useruploadpolicy –u user_name –p policy_name –f policy_file.txt \
     --delegate=account_name
```

The `euare-userlistpolicies` command lists the names of a policies associated with a user. It has the following syntax:

```
euare-userlistpolicies –u <user_name> <--delegate=account_name>
```

This command can be run by the administrator of the account, or if using the `--delegate` option, by a user in the eucalyptus account (a cloud administrator). If run by a cloud administrator, it would display the polices associated with the user in the account listed in the `--delegate` option. While not shown above, the content of the policy would also be displayed if the `–v` option was included.

There are additional options to limit the number of policies displayed on the screen and to paginate the display results. For more information, see the online help page for the command.

The `euare-usergetpolicy` command displays the contents of a single named policy associated with the named user, but has no pagination options. It has the following syntax:

```
euare-usergetpolicy –u <user_name>   \
     -p <policy_name> <--delegate=account_name>
```

This command can be run by the administrator of the account, or if using the `--delegate` option, by a user in the eucalyptus account (a cloud administrator). If run by a cloud administrator, it would display the polices associated with the user in the account listed in the `--delegate` option.

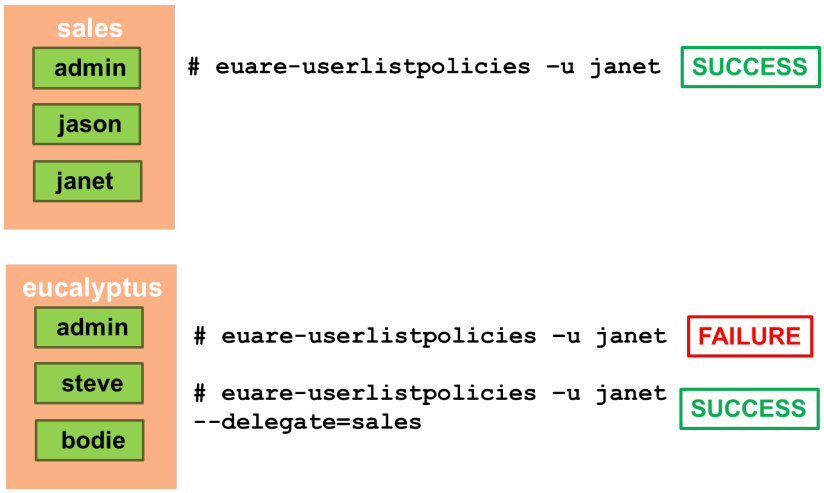
The `euare-userdelpolicy` command deletes the named policy from the named user. It has the following syntax:

```
euare-userdelpolicy –u <user_name>   \
     -p <policy_name> <--delegate=account_name>
```

This command can be run by the administrator of the account, or if using the `--delegate` option, by a user in the eucalyptus account (a cloud administrator). If run by a cloud administrator, it would delete the polices associated with the user in the account listed in the `--delegate` option.

Delegate Option

The delegate option can only be used by a cloud administrator (a member of the *eucalyptus* account). It is used by a cloud administrator to view and edit information associated with other non-*eucalyptus* accounts.



In the first example above the *admin* user of the *sales* account uses the `euare-userlistpolicies` command to list all the policies associated with the user *janet*. Because the command was issued by a user in the *sales* account, the command searches the *sales* account. Because there is a user named *janet* in the *sales* account, the command succeeds.

In the second example there are three cloud administrators in the *eucalyptus* account. If any one of them uses the first command listed, the command will fail. That is because the command will search the *eucalyptus* account for a user named *janet*. The second command in this example succeeds because of the `--delegate` option. The `--delegate` option forces the command to search in the *sales* account for the user *janet* rather than search the local *eucalyptus* account.

Access Management Controls

Access control is available on two levels:

- Per-resource permissions on Walrus (S3) buckets and Walrus (S3) images can be set to control resource access across accounts
- Policies can be used to control access within an account

Permissions	Attached To	Scope	Service	Set By
S3 bucket ACL	specific resource	across accounts	S3	`S3curl.pl`
S3 image permission	specific resource	across accounts	EC2	`euca-upload-bundle --acl`
IAM	users/groups	inside account	all	policies

When a request is received for a bucket, Eucalyptus checks the corresponding bucket access controls to verify whether the requester has the necessary access permissions. When you create a bucket, S3 creates a default set of permission that grants the bucket owner full control over the resource. S3 (Walrus) bucket access control is set by using API calls that update a bucket's access control list. One way to update the access controls associated with a bucket is to use the S3curl.pl program.

Image objects inside a bucket also have permissions. These permission can be controlled by using the `--acl` option of the `euca-upload-bundle` command. The command `--acl` *acl* has the following characteristics:

- Valid Values: `public-read | aws-exec-read`

- Default: `aws-exec-read`
- Example: `--acl public-read`

Note: The `aws-exec-read` value gives read permission to the Eucalyptus process that launches EMIs. That means that the user who uploads the image has read access to it. If the user is a cloud administrator, then everyone has read access to it.

Evaluate Permissions

Permission evaluation has to accommodate the fact that, for some resources, there are two-levels of resource controls:

1. If the user is a cloud administrator, access is granted.
2. If the user is an account administrator, access is granted (to account resources).
3. If the user is a normal user and denied access at the resource-level, access is denied.
4. If the user is a normal user and is allowed access at the resource-level, evaluate all EIAM policies in effect for the user:
   - If there is no policy match, access is denied (implicit deny).
   - If the user is specifically denied, access is denied.
   - If a user is specifically granted, access is granted.
   - If a user is specifically granted *and* denied, access is granted.

Note: Allow policies always override deny policies. The most permissive policy prevails.

Quotas

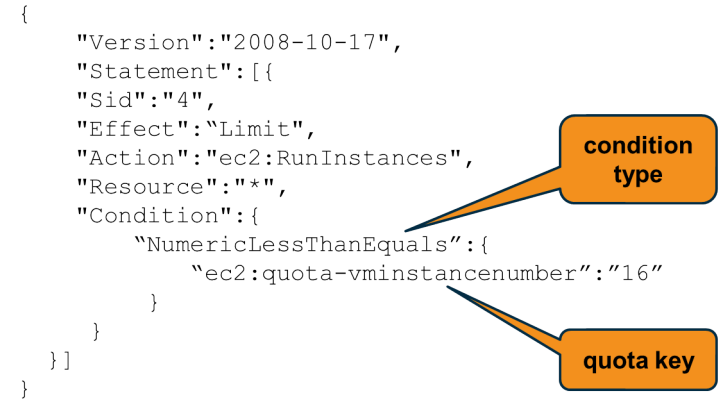
There are no quota limits in a public cloud as they would be contrary to the business goals of the public cloud owner. A public cloud revenue stream is based on users using the cloud resources. The more resources the users use, the more revenue the public cloud provider receives. Also, from a individual user perspective, the public cloud resources might appear unlimited. A private cloud is different in that there are limits on the physical resources. Because resources are more limited in the private cloud, the ability to enforce resource quotas on users makes more sense.

For this reason Eucalyptus extended the Amazon IAM API to include resource quotas. The policy statement *Effect* was extended to include *Limit* along with the normal Allow and Deny keywords. Where Limit appears as the Effect, it is a quota statement and not a permissions statement. While condition blocks are optional in permission statements, they are required in quota statements. However, the only condition will always be NumericLessThanEquals. The condition block will include the actual quota type and settings.

A quota statement also has *Action* and *Resource* fields, which are used to match specific requests. The actual quota type and value are specified using special quota keys listed in the condition. Only the condition type NumericLessThanEquals can be used with quota keys.

Quota Example

This quota statement, attached to a user, would limit them to running a maximum of 16 instances at a time in the cloud:



Note: The only condition type allowed in quota statements is NumericLessThanEquals.

Quota Keys

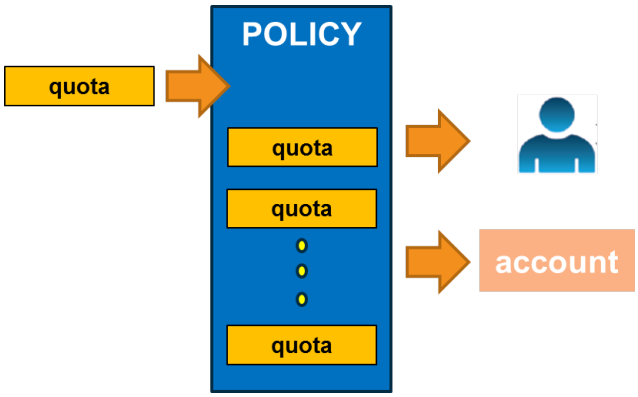
The following tables identify quota key names and their descriptions. The *keyname* is the string that is entered into the condition block in the quota policy. The scope determines whether the quota can be applied to an account or an individual user, or either.

Keyname	Description	Scope
s3:quota-bucketnumber	Number of S3 buckets	account and user
s3:quota-bucketobjectnumber	Number of objects in each bucket	account and user
s3:quota-bucketsize	Size of bucket in MB	account and user
s3:quota-buckettotalsize	Total size of all buckets in MB	account and user
ec2:quota-addressnumber	Number of elastic IPs	account and user
ec2:quota-imagenumber	Number of EC2 images	account and user

Keyname	Description	Scope
ec2:quota-snapshotnumber	Number of EC2 snapshots	account and user
ec2:quota-vminstancenumber	Number of EC2 instances	account and user
ec2:quota-volumenumber	Number of EC2 volumes	account and user
ec2:quota-volumetotalsize	Number of total volume size in GB	account and user
iam:quota-groupnumber	Number of IAM groups	account only
iam:quota-usernumber	Number of IAM users	account only

Quota Rules

Quotas can be attached to both users and accounts.



Some quota keys only apply to accounts.

Account quotas are attached to the *admin* user of the account.

Note:

Quotas attached to groups will take no effect.

Quota Evaluation

A user might be affected by multiple quota limits.

1. If the user is a cloud administrator, there are no limits
2. If the user is an account administrator, reject any request that exceeds an account-level quota .
3. If the user is a normal user, reject any request that exceeds any account or user-level quota .

Note: System properties - displayed when you run the `euca-describe-properties command` - override quotas. For example, the system property `walrus.storagemaxbucketsizeinmb` sets a cloud-wide hard limit for the maximum size of a Walrus bucket. The quota key `s3:quota-bucketsize` cannot increase this size for an individual user or an account.

Quota Account Policies

Account quota policies can be uploaded from an external file using the command `euare-accountuploadpolicy`. The syntax is as follows:

```
euare-accountuploadpolicy –a <account_name>  \
     -p <policy_name> -f <policy_filename>
```

The `euare-accountlistpolicies` command lists the names of a quota policies associated with an account. The syntax for this command is:

```
euare-accountlistpolicies –a <account_name>
```

There are additional options to limit the number of policies displayed on the screen and to paginate the display results. For more information, see the online help page for the command.

The `euare-accountgetpolicy` command displays the contents of a single named quota policy associated with the named account, but has no pagination options. The syntax for this command is:

```
euare-accountgetpolicy –a <account_name> -p <policy_name>
```

The `euare-accountdelpolicy` command deletes the named quota policy from the named account. The syntax for this command is:

```
euare-accountdelpolicy –a <account_name> -p <policy_name>
```

Note: None of the quota account policy commands listed have the `--delegate` option.

Managing Accounts, Groups, and Users

This section describes management operations on accounts, groups, and users from both the Eucalyptus Administrator Console and the command line.

It is not exhaustive, but does include the most common operations.

Note: For more information, see the *Eucalyptus Command Line Interface Reference Guide* at *http://www.eucalyptus.com/docs*.

Adding Accounts

There are two different ways that a new account can be added to the cloud.

A user could connect to the Eucalyptus Administrator Console and use it to request a new account for a department or division within their company or organization. The cloud administrator would review the request and either approve or deny it.

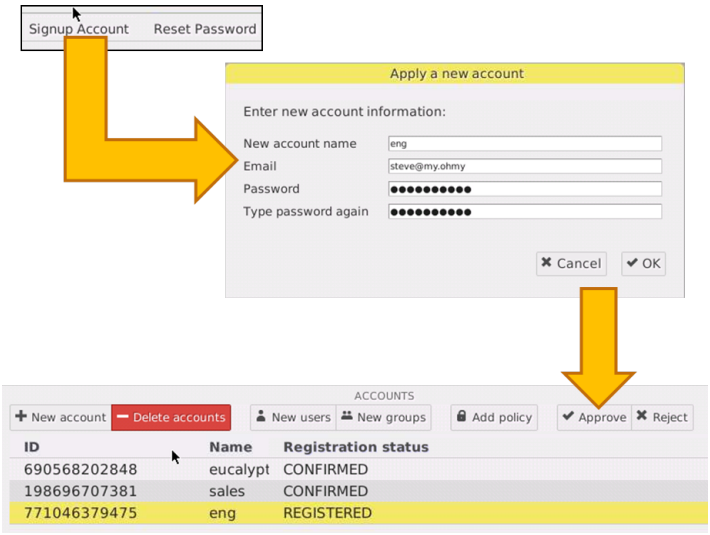
In the other method, the cloud administrator creates a new account and then tells the user who will be the account *admin* user how to connect to the cloud and log in as that user. This lesson describes both methods.

Note: If a Eucalyptus 2 cloud is upgraded to a Eucalyptus 3 cloud, any existing Eucalyptus 2 users will be converted into Eucalyptus 3 accounts.

User-Initiated

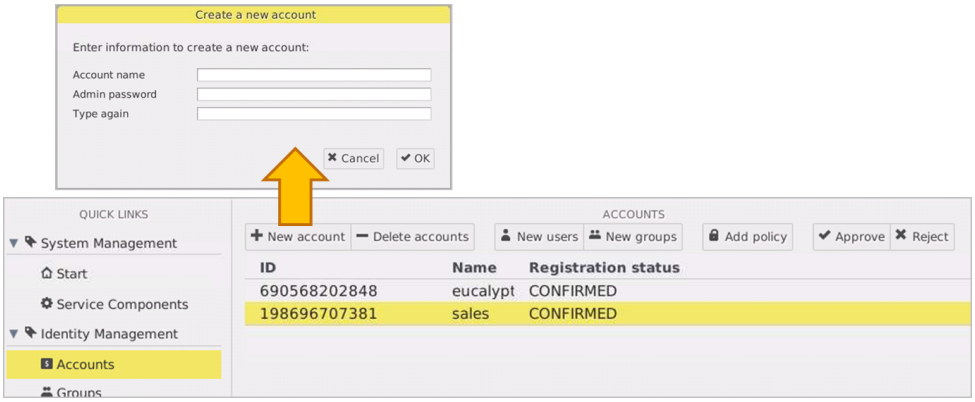
In this method the user uses the Eucalyptus Administrator Console to request a new account from the cloud administrator. The cloud administrator could either see the request via email or notice the pending request in the Administrator Console. Either way, the cloud administrator would either approve or deny the request in the Administrator Console. An email is sent to the user with either the denial or with the information about how to access

the cloud and log in. Approved accounts are listed with a registration status of CONFIRMED in the Administrator Console. Pending account requests are listed with a status of REGISTERED in the Administrator Console.



Administrator-Initiated

To create an account requires the user to be a cloud administrator. Accounts can be created using either the Eucalyptus Administrator Console or the command line. Both are illustrated below. To create an account in the Administrator Console select **Accounts** in the QUICK LINKS panel and click the **New account** button. Once you fill out the form the new account is created. When you create a new account using the Administrator Console you will need to supply the password for the *admin* user of the account.



The command-line syntax is as follows:

```
euare-accountcreate -a <account_name>
```

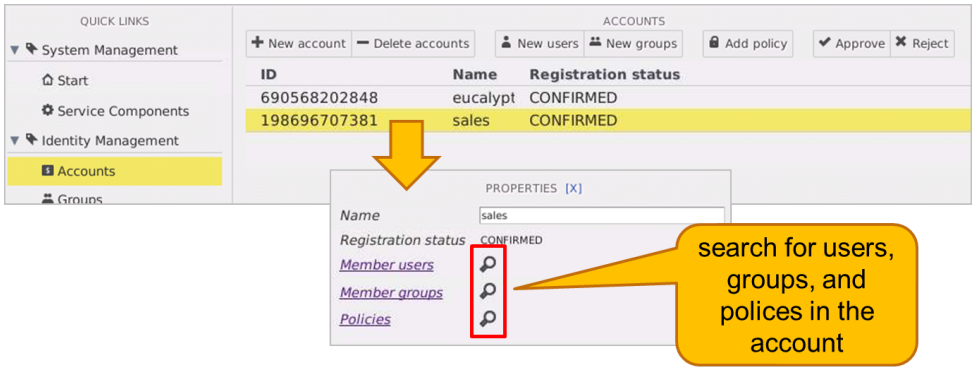
For example, `euare-accountcreate –a sales` would create the *sales* account.

When you add a new account, you cannot specify a specific path for the *admin* user for the account. To specify a path other than the default `/`, once the account is created go to the Administrator Console and select the *admin* user and change the path in the PROPERTIES panel.

Viewing Accounts

Accounts can be viewed and searched from the Eucalyptus Administrator Console or command line.

To view accounts in the Administrator Console select **Accounts** in the QUICK LINKS panel and the list of existing accounts appears in the main panel. You can click any account listed to get additional information about that account including member users, member groups, and attached policies.



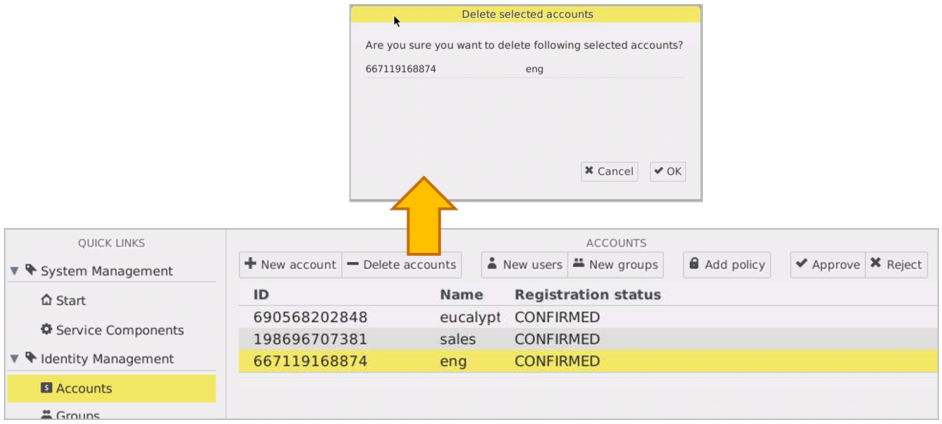
The `euare-accountlist` command returns the names of all the accounts. There is no `--delegate` option for this command.

The `euare-accountgetsummary` command returns the number of groups and users in an account, but does not list the group names or user names. The `--delegate` option exists, but can only be run by a cloud administrator (user in the *eucalyptus* account). It allows the user to display information about accounts other than the *eucalyptus* account.

Deleting Accounts

Accounts can be deleted from the Eucalyptus Administrator Console or the command line.

To delete an account from the Administrator Console select **Accounts** in the QUICK LINKS panel, select the account you wish to delete, and click the **Delete accounts** button. Once you confirm the operation, the account and all users, keys, certificates, passwords, and groups in the account will be deleted.



The `euare-accountdel` command deletes accounts. It has the following syntax:

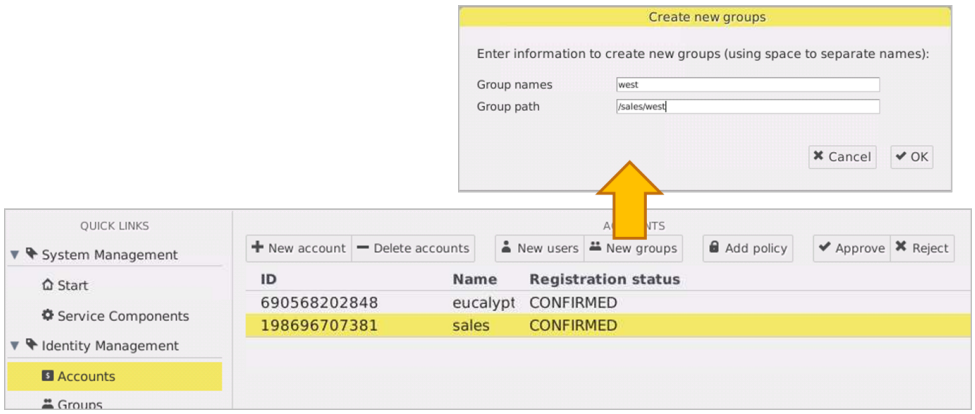
```
euare-accountdel –r –a <account_name>
```

The `–r` option recursively deletes the account. This means that all users and groups in the account are deleted along with the account. This includes any user keys, certificates, and login profiles.

Adding Groups

Groups can be added to an account using the Eucalyptus Administrator Console or command line.

To add a group to an account from the Administrator Console select **Accounts** in the QUICK LINKS panel, select the account in the list in the main panel, and click the **New groups** button. Provide the group name and an optional path. If no path is given it will default to /.



The `euare-groupcreate` command creates a group in an account. It has the following syntax:

```
euare-groupcreate –g group_name –p path <--delegate=account_name>
```

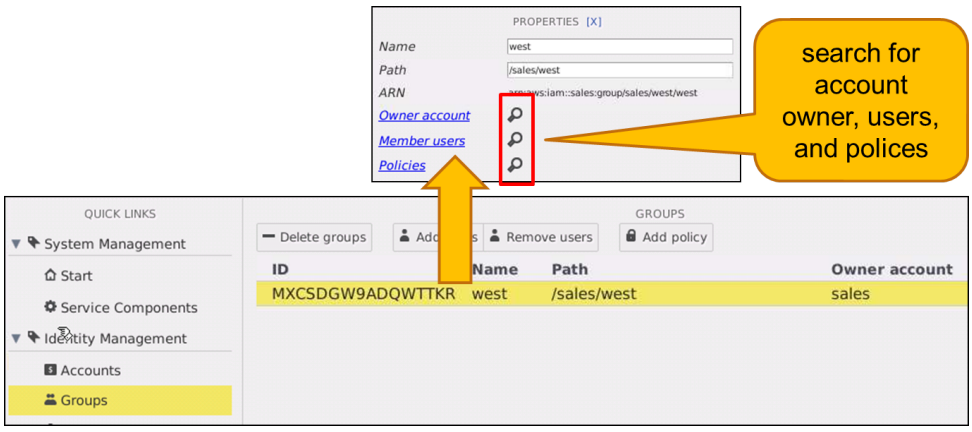
This command can be run by the administrator of the account, or if using the `--delegate` option, by a user in the *eucalyptus* account (a cloud administrator). If run by a cloud administrator, the group would be created in the account listed in the `--delegate` option.

The `–p` argument is optional and will default to `/` if not included.

Viewing Groups

Groups can be viewed using the Eucalyptus Administrator Console or command line.

To view all groups in the cloud from the Administrator Console, select **Groups** in the QUICK LINKS panel, and the list of groups appears in the main panel.



There are different commands used for listing groups by path, listing group users, and listing group policies. These commands can be run by the administrator of the account, or if using the `--delegate` option, by a user in the *eucalyptus* account (a cloud administrator).

The `euare-grouplistbypath` command lists the groups in an account. It has the following syntax:

```
euare-grouplistbypath <–p path_prefix> <--delegate=account_name>
```

Note: The `–p` argument is optional and will default to `/` if not included. This option can be used to filter the list to only those groups with the specified `path_prefix`.

If `euare-grouplistbypath` is run by a cloud administrator, the groups listed would be those in the account listed in the `--delegate` option.

The `euare-grouplistusers` lists the users in a group. It has the following syntax:

```
euare-grouplistusers –g group_name <--delegate=account_name>
```

If `euare-grouplistusers` is run by a cloud administrator, the group whose users are listed would be a group in the account listed in the `--delegate` option.

The `euare-grouplistpolicies` command lists the policies associated with an account. It has the following syntax:

```
euare-grouplistpolicies –g group_name <--delegate=account_name>
```

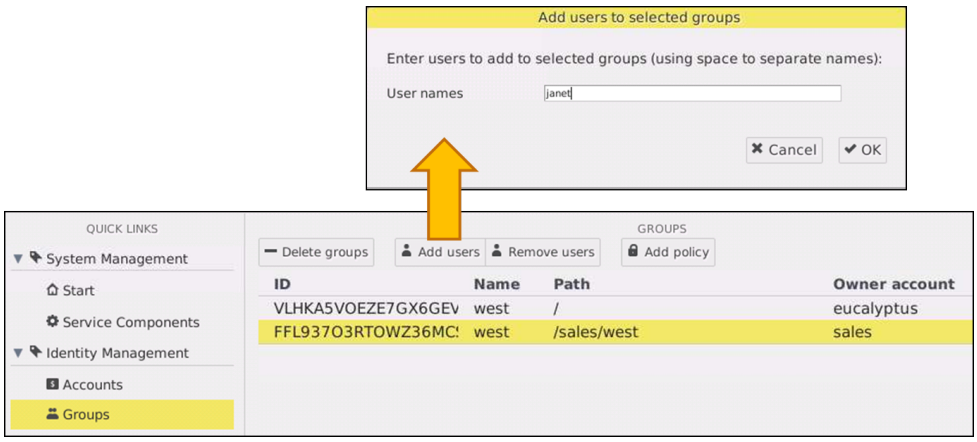
If `euare-grouplistpolicies` is run by a cloud administrator, the group whose policies are listed would be a group in the account listed in the `--delegate` option.

Note: There are additional options to limit the number of groups displayed on the screen and to paginate the display results. For more information, see the online help page for the command.

Adding Users to Groups

Users can be added to a group using the Eucalyptus Administrator Console or command line.

To add users to a group in the Administrator Console, select **Groups** in the QUICK LINKS panel, select the group from the list in the main panel, then click the **Add users** button to add a user.



The `euare-groupadduser` command add a user to the named group. It has the following syntax:

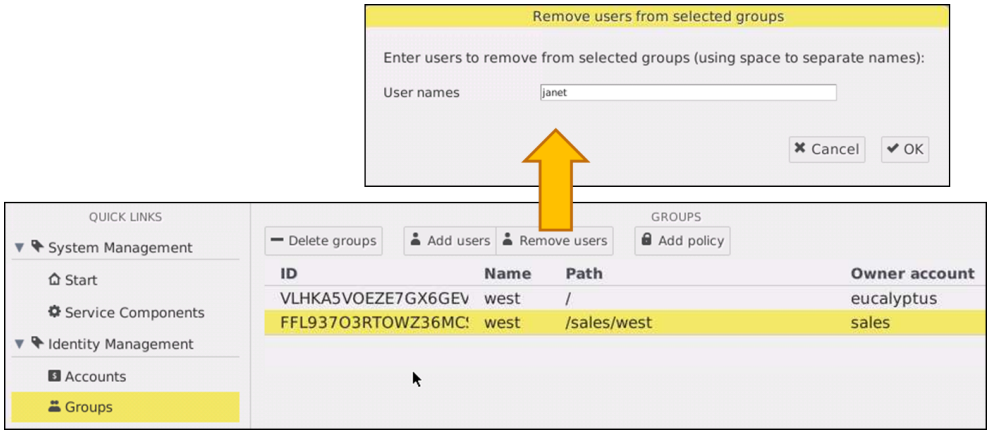
```
euare-groupadduser  –u  user_name  –g  group_name  \
     <--delegate=account_name >
```

The command can be run by the administrator of the account, or if using the `--delegate` option, by a user in the *eucalyptus* account (a cloud administrator). If the command is run by a cloud administrator, the group to which a user was added would be a group in the account listed in the `--delegate` option.

Removing Users from Groups

Users can be removed from a group using the Eucalyptus Administrator Console or command line.

To remove a user from a group from the Administrator Console, select **Groups** in the QUICK LINKS panel, select the group in the main panel, then click the **Remove users** button.



The `euare-groupremoveuser` command removes a user from the named group. It has the following syntax:

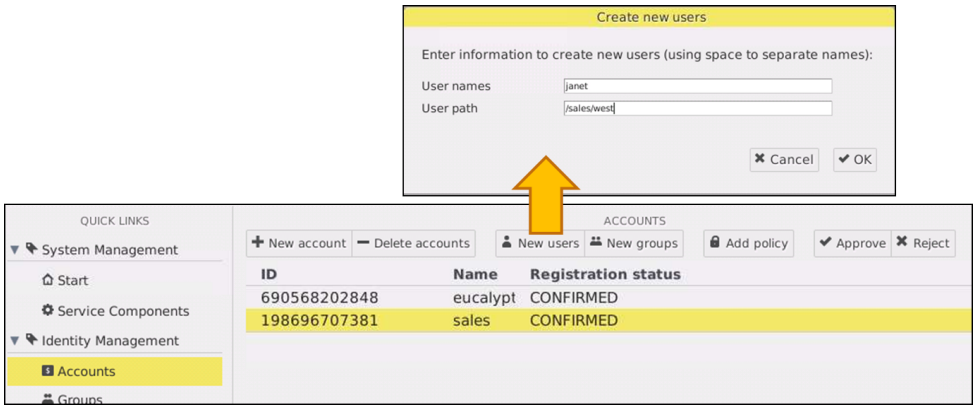
```
euare-groupremoveuser –u user_name –g group_name  \
     <--delegate=account_name>
```

The command can be run by the administrator of the account, or if using the `--delegate` option, by a user in the *eucalyptus* account (a cloud administrator). If the command is run by a cloud administrator, the group to which a user is removed would be a group in the account listed in the `--delegate` option.

Adding Users to an Account

Users can be added to an account from the Eucalyptus Administrator Console or command line.

To add users to an account from the Administrator Console, select **Accounts** in the QUICK LINKS panel, select the account you wish to add the user to, and then click the **New users** button. Once you complete the form the user is added.



The `euare-usercreate` command creates a new user in an account. It has the following syntax:

```
euare-usercreate -u user_name <-g group_name> <-p path>  \
     <--delegate=account_name>
```

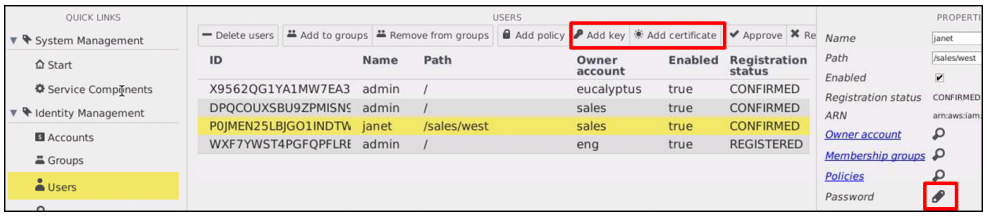
This command can be run by the administrator of the account, or if using the `--delegate` option, by a user in the *eucalyptus* account (a cloud administrator). If run by a cloud administrator, the user would be created in the account listed in the `--delegate` option.

The user can optionally be added to an existing group using the `-g` option.

The path defaults to `/` if not specified with the `-p` option.

User Password and Credentials - Administrator Console

The Eucalyptus Dashboard can be used to create a user login, password, access keys, and certificate.



To create an initial login password, click the **Password** icon.

To create access keys and a certificate:

1. Click **Add key** and **OK**.
2. Click **Add certificate**, copy/paste the certificate content, and click **OK**.

Note: While the Eucalyptus Administrator Console will create access keys for the user, it will not create a certificate for the user. For certificates, the Administrator Console expects you to copy and paste an existing certificate in the Add certificate dialog box. .

Alternatively, the user can log in and download their credentials, because keys and a certificate are automatically generated in response to a request to download credentials in the Administrator Console.

Note: If there is no pre-existing credential and you would like the cloud to generate a credential for the user, use the command line (`euare-usercreatecert`) rather than the Administrator Console.

User Password and Credentials - Command-Line

User login, password, access keys, and certificates can be created from the command line.

The `euare-useraddloginprofile` is used to create a login password for a user account. It has the following syntax:

```
euare-useraddloginprofile –u user_name    \
     –p password <--delegate=account_name>
```

This command can be run by the administrator of the account, or if using the `--delegate` option, by a user in the *eucalyptus* account (a cloud administrator). If run by a cloud administrator, the password would be created for the user in the account listed in the `--delegate` option.

The `euare-useraddkey` is used to create access keys for a user account. It has the following syntax:

```
euare-useraddkey –u user_name <--delegate=account_name>
```

This command can be run by the administrator of the account, or if using the `--delegate` option, by a user in the *eucalyptus* account (a cloud administrator). If run by a cloud administrator, the access keys would be created for the user in the account listed in the `--delegate` option.

Note: The `–k` option can be added to the `euare-usercreate` command to generate access keys for the user at the time the account is created.

The `euare-usercreatecert` is used to create a certificate for a user. It has the following syntax:

```
euare-usercreatecert –u user_name <--delegate=account_name>
```

This command can be run by the administrator of the account, or if using the `--delegate` option, by a user in the *eucalyptus* account (a cloud administrator). If run by a cloud administrator, the certificate would be created for the user in the account listed in the `--delegate` option.

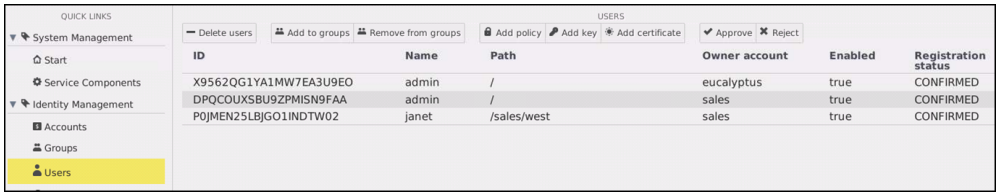
There are other commands to list, modify, and delete a user login password, access keys, and certificate. They include:

- `euare-usergetloginprofile`,
- `euare-usermodloginprofile`,
- `euare-userdelloginprofile`,
- `euare-userlistkeys`,
- `euare-usermodkey`,
- `euare-userdelkey`,
- `euare-userlistcerts`,
- `euare-usermodcert`, and
- `euare-userdelcert`.

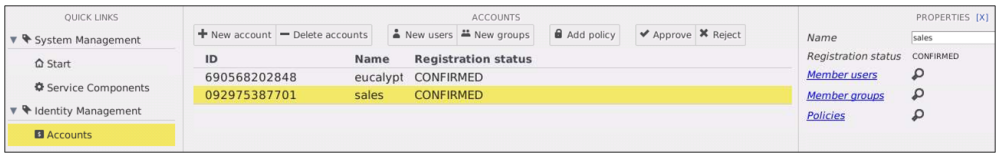
See the Web-based Eucalyptus documentation or online command help pages for more information.

Viewing Users

Users from every account can be viewed from the Eucalyptus Administrator Console by selecting **Users** in the QUICK LINKS panel.



To view users in a specific account select **Accounts** in the QUICK LINKS panel, then select an account, and click the search icon next to **Member users**.



The `euare-userlistbypath` command can also be used to list the users in an account. It has the following syntax:

```
euare-userlistbypath <-p /path> <--delegate=account_name>
```

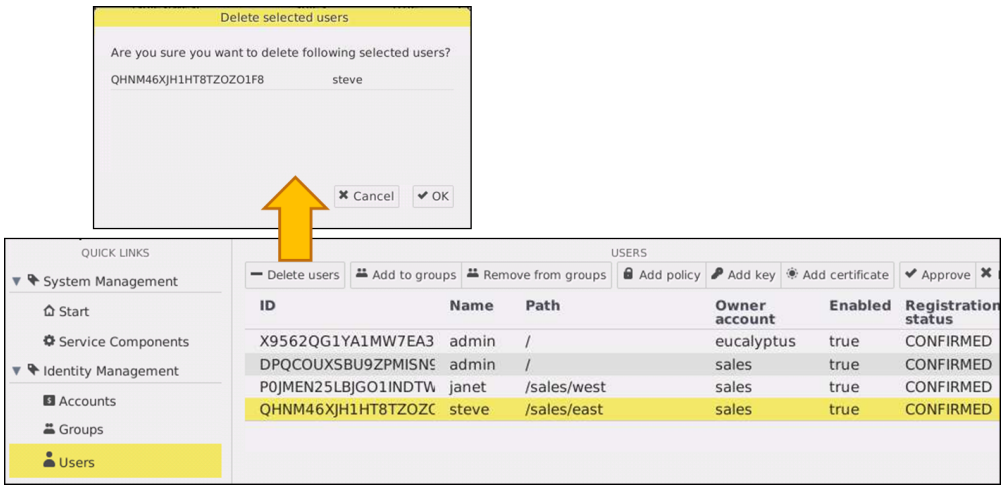
This command can be run by the administrator of the account, or if using the `--delegate` option, by a user in the *eucalyptus* account (a cloud administrator). If `euare-userlistbypath` is run by a cloud administrator, the users listed would be those in the account listed in the `--delegate` option.

The `–p` argument is optional and will default to `/` if not included. This option can be used to filter the list to only those users with the specified `/path` prefix.

There are additional options to limit the number of users displayed on the screen and to paginate the display results. For more information, see the online help page for the command.

Deleting Users

Users from any account can be deleted using the Eucalyptus Administrator Console or the command line. To delete a user from the Administrator Console select **Users** from the QUICK LINKS panel, select the user to delete from the list, and then click the **Delete users** button.



The `euare-userdel` command can be used to delete users from the command line. It has the following syntax:

```
euare-userdel –r –u <user_name> <--delegate=account_name>
```

The `-r` option recursively removes the users credentials, policies, and login profile, and also removes the user from any groups.

There is also a `–p` preview option that shows what the command would do, but without actually doing it.

Lab - Managing Eucalyptus Accounts, Users, and Groups

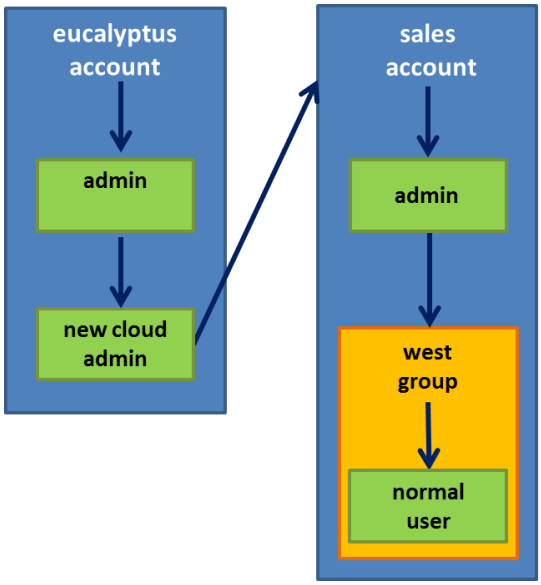
In this lab exercise you will manage accounts, users, and groups using the Eucalyptus Administrator Console.

You will log in to the *eucalyptus* account as the user *admin* and add a new cloud administrator user. After logging in as this new cloud administrator, you will add a new *sales* account to the cloud. You will also add a new group named *west* to the *sales* account.

Each new account created automatically includes a new *admin* user for that account. You will log in as *admin* to the *sales* account and add a *normal* (non-administrative) user to the *sales* account. You will also add the *normal* user to the *west* group in the *sales* account.

You will finish by logging in to the Administrator Console as the *normal* user in the *sales* account and downloading the user's credentials.

The following diagram describes relationships between the accounts, groups, and users in this lab.



Lab Objectives:

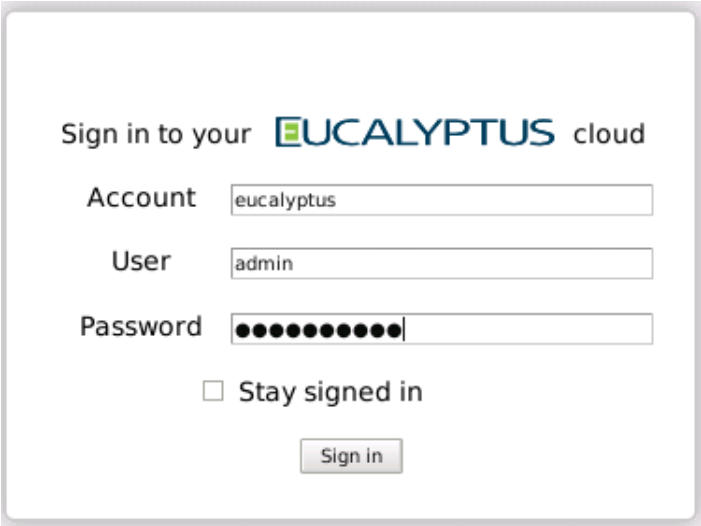
- Add a new administrative user in the *eucalyptus* account
- Log in and test the new administrative user
- Create a non-administrative user
- Log in as a non-administrative user

Add a new administrative user in the eucalyptus account

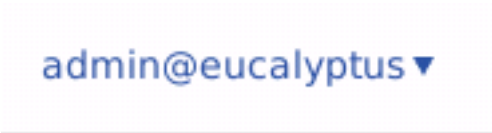
Desktop

In this section of the lab you will view the existing users in the *eucalyptus* account. Then you will add a new administrative user to the *eucalyptus* account.

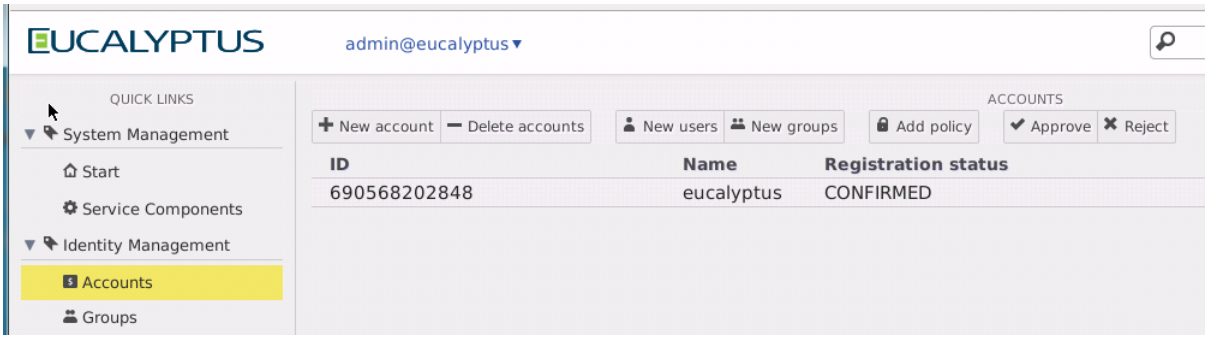
1. From the Debian desktop, open the browser and use the URL https://<front_end_public_IP>:8443 to access the Eucalyptus Administrator Console. Log in to the `eucalyptus` account as the user `admin` with a password of `passwordNN`, where *NN* is the number of your student pod.



2. Note the current user name and account name above the main panel in the Administrator Console.

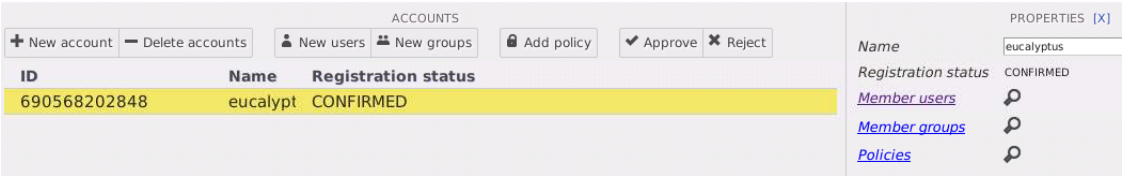


3. In the Administrator Console, click **Accounts** in the QUICK LINKS panel.

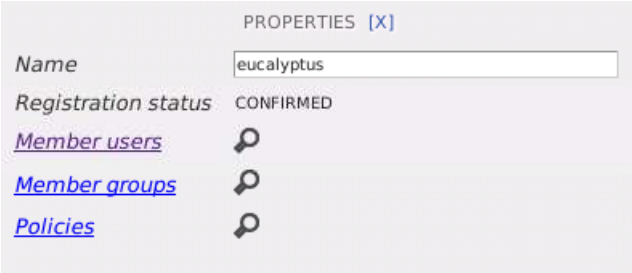


Which accounts exist?

4. Click the **eucalyptus** account in the main panel. The PROPERTIES panel appears.

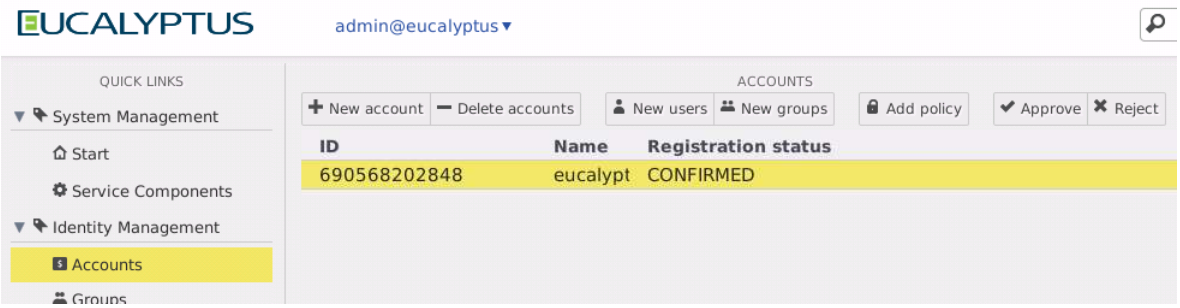


5. Click **Member users** in the PROPERTIES panel to display the users in the *eucalyptus* account.

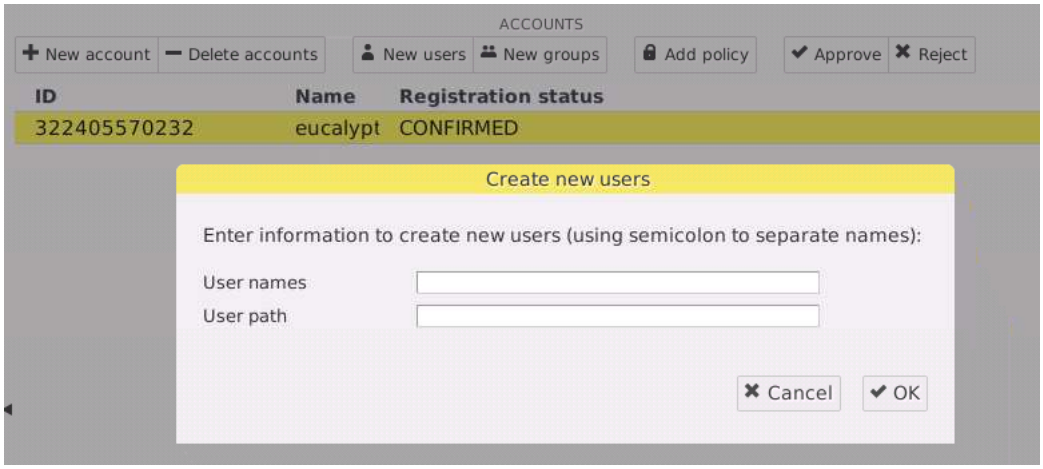


What is the only user in the *eucalyptus* account?

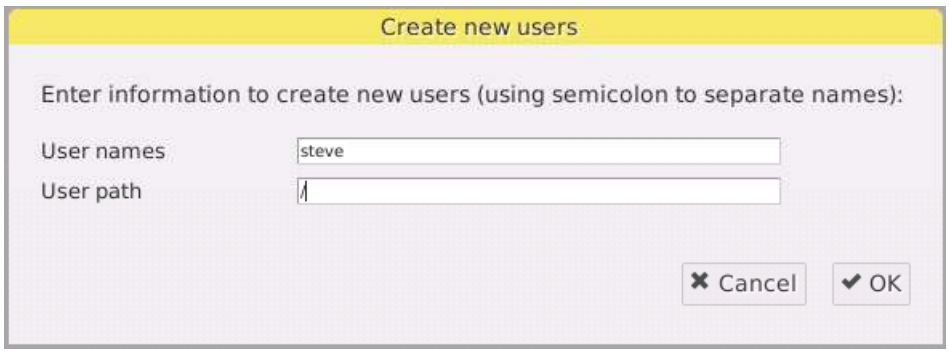
6. Click **Accounts** in the QUICK LINKS panel again. Then click the *eucalyptus* account.



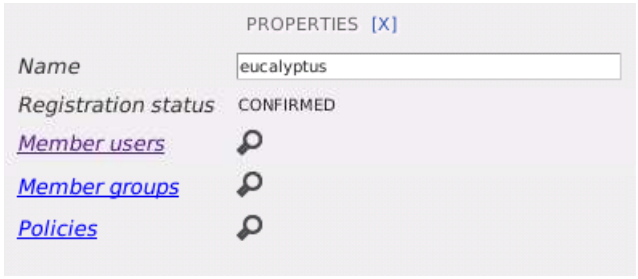
7. Click the **New users** button to add a new cloud administrator user to the *eucalyptus* account. The Create new users window appears.

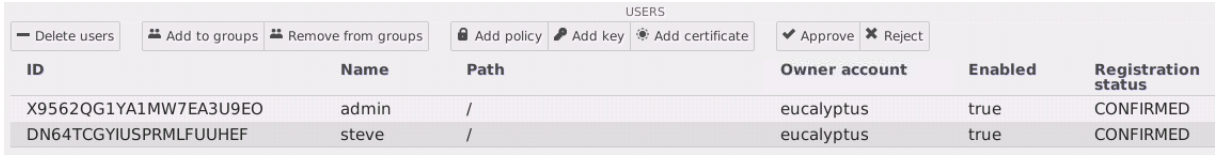


8. Enter your first name in the **User names** text box and type / in the **User path** text box. Click **OK**.



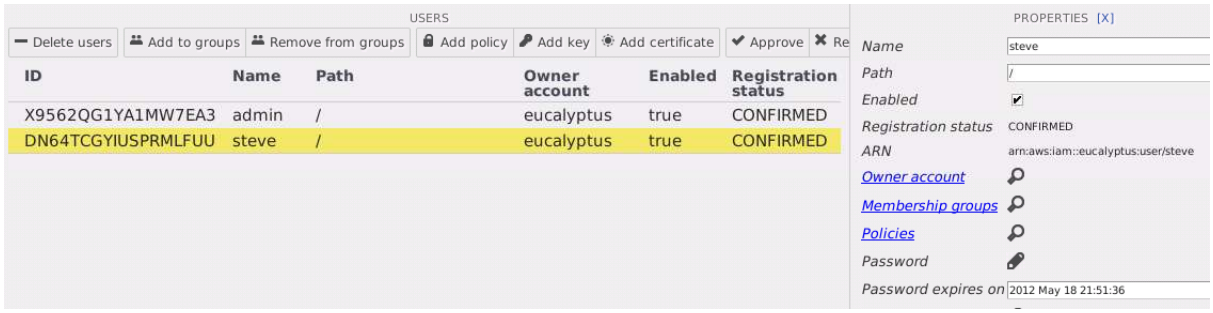
9. With the *eucalyptus* account highlighted, click **Member users** again.



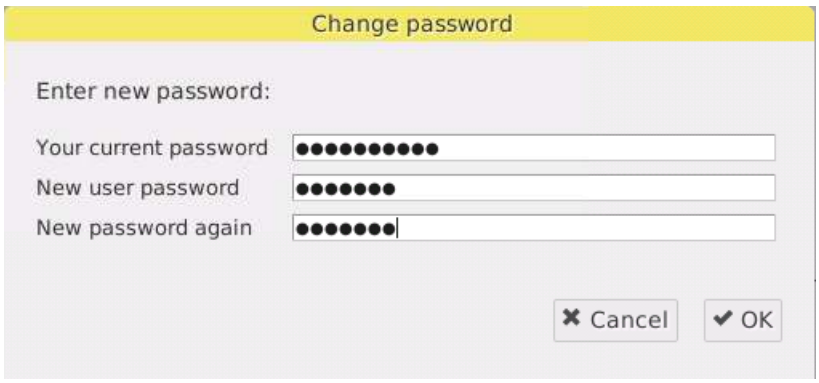


Which users are now members of the *eucalyptus* account?

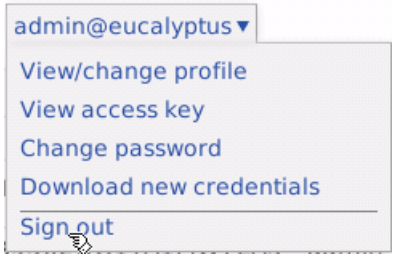
10. Click your new user in the main panel, and then click the pencil icon next to **Password** in the PROPERTIES window in order to create an initial login password for the new user.



11. When the Change password window opens, enter your current *admin* password (it should be password*NN)* and then enter `newuser` as the initial password of the new cloud administrator user. Click **OK**.



12. To sign out of the Administrator Console, click the current login name above the main panel and select **Sign out** on the drop-down menu.

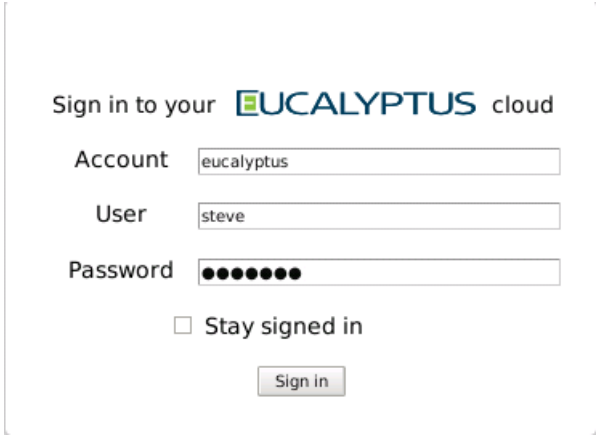


Log in and test the new administrative user

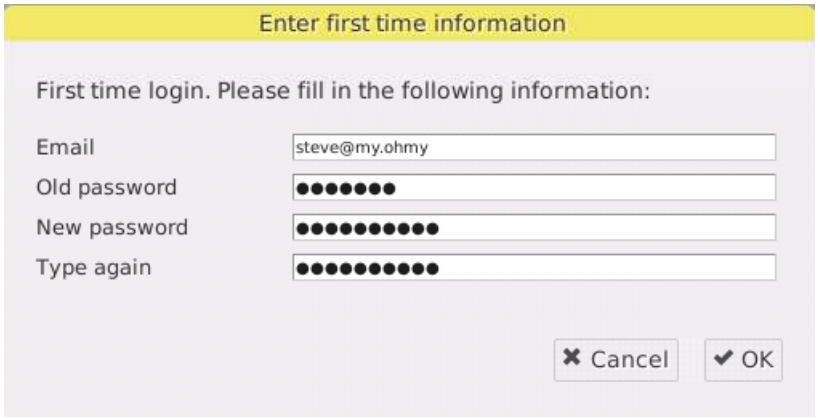
Desktop

In this section of the lab you will log in as the new cloud administrator user and perform some administrative tasks. These tasks will include adding a new *sales* account and adding a new group *west* to the new *sales* account.

1. Log in to the Eucalyptus Administrator Console as the new cloud administrator user in the `eucalyptus` account. Remember the user is your first name with an initial password of `newuser`. Click **Sign in**.

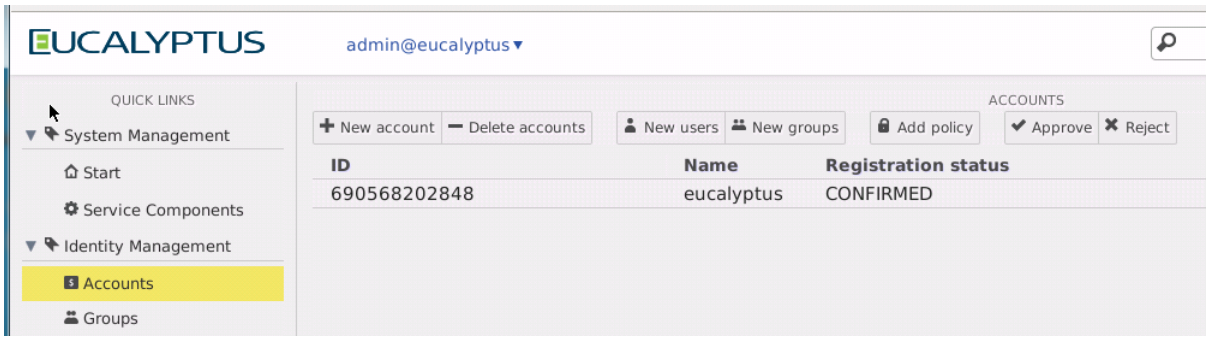


2. Complete the initial log in by entering your information in the Enter first time information window. Enter any email address that you wish. Enter the old password, and then for the new password use the string `passwordNN`, where *NN* is the number of the student pod that you are assigned. Click **OK**.

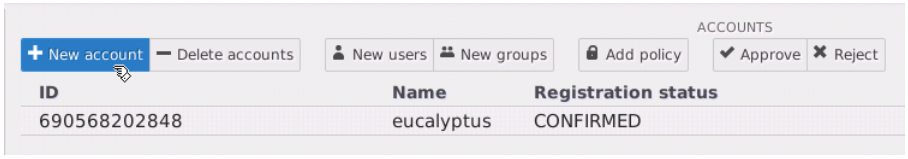


3. Note the current user name and account name above the main panel in the Administrator Console.

4. Click **Accounts** in the QUICK LINKS panel.



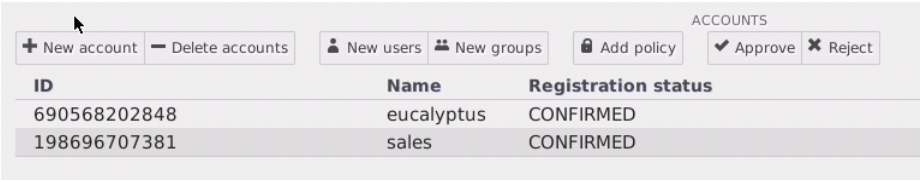
5. Click the **New account** button.



6. In the Create a new account window, type `sales` as the new **Account name**. For the *admin* user's password, use the initial password of `newaccount`. Click **OK**.

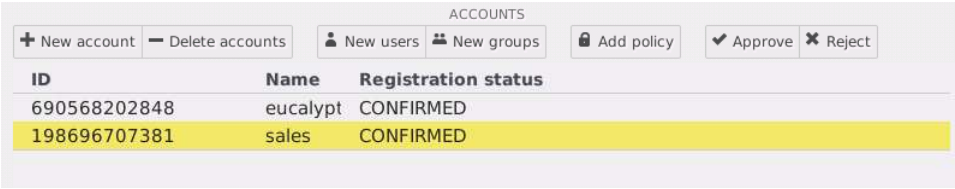
Did it work?

7. Note that the new account appears in the main panel in the list with the *eucalyptus* account.



Note: Notice that the new cloud administrator user that you created in the *eucalyptus* account had sufficient administrative permissions to be able to successfully create a new account in your cloud.

8. Click the **sales** account in the main panel to highlight it, and then click the **New groups** button.

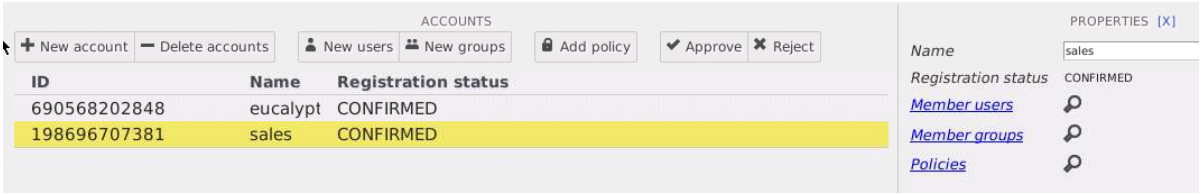


9. In the Create new groups window, type `west` for the **Group name** and `/sales/west` for the **Group path**. Click **OK**.

Did it work?

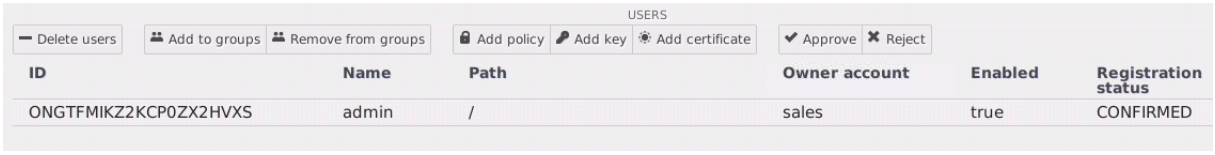
Note: Notice once again that your new cloud administrator user had sufficient permissions to create a new group in your new account.

10. With the **sales** account still selected in the main panel of the Administrator Console, click **Member users** in the PROPERTIES panel. This will display only users in the *sales* account.



Note: Clicking **Member groups** would have displayed only groups in the *sales* account.

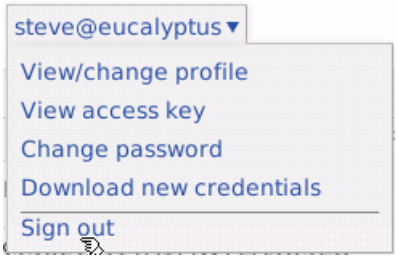
11. Which user was automatically created in the *sales* account?



12. Click **Users** in the QUICK LINKS panel.

Note: **Users** in the QUICK LINKS panel displays all users in the cloud regardless of which account that they belong to.

13. Sign out of the Administrator Console.



Create a non-administrative user

Desktop

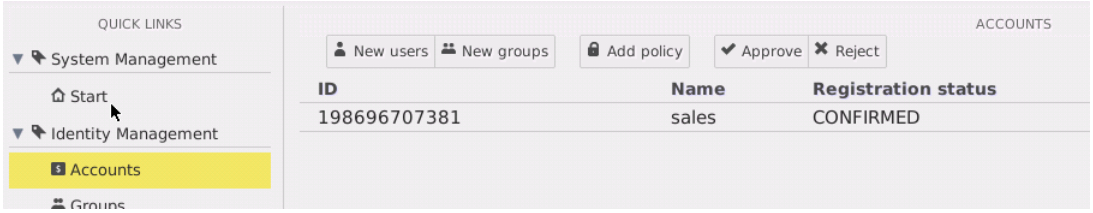
In this section of the lab you will create a non-administrative (a normal user) user in the *sales* account. You will name this user *normal*. You will also add this user to the group *west* in the *sales* account. However, rather than perform this task as a cloud administrator (which would work), you will perform this task as an account administrator.

1. Log in to the Eucalyptus Administrator Console as the *admin* of the *sales* account. The initial password should be `newaccount`. Click **Sign in**.

2. Enter the *admin* user's first time log in information. Enter any email account that you wish, the old password should be `newaccount`, and set the new password to the standard `passwordNN` password, where *NN* is the number of the pod that you are assigned. Click **OK**.

3. Note the current log in name and account name at the top of the Administrator Console.

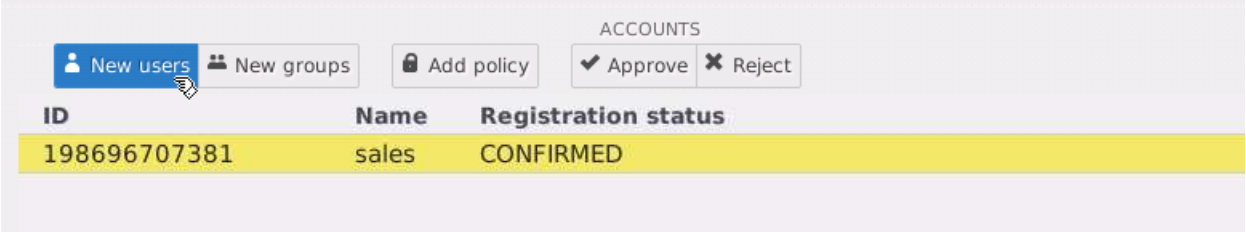
4. Click **Accounts** in the QUICK LINKS panel.



What is the only account that appears based on the permissions of this user?

Note: Notice the *admin* user for the *sales* account does not have the permissions to view the *eucalyptus* account.

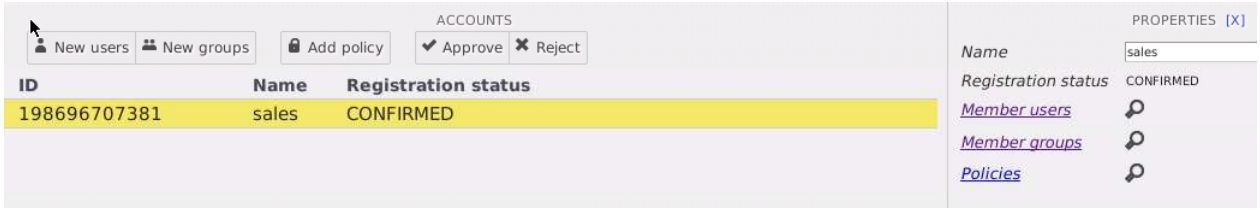
5. Click the **sales** account in the main panel and then click the **New users** button.



6. In the Create new user window, create a new user named `normal` and type `/sales/west` for the user's path. Click **OK**.

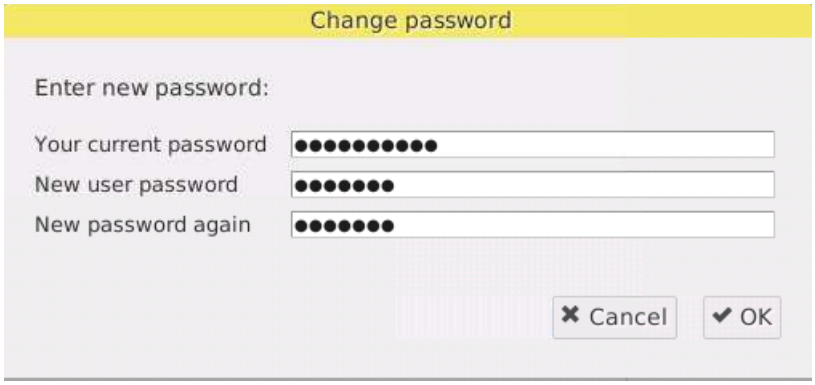
Did the *admin* user for the *sales* account have sufficient permissions to perform this operation?

7. With the **sales** account still selected, click **Member users** in the PROPERTIES panel.

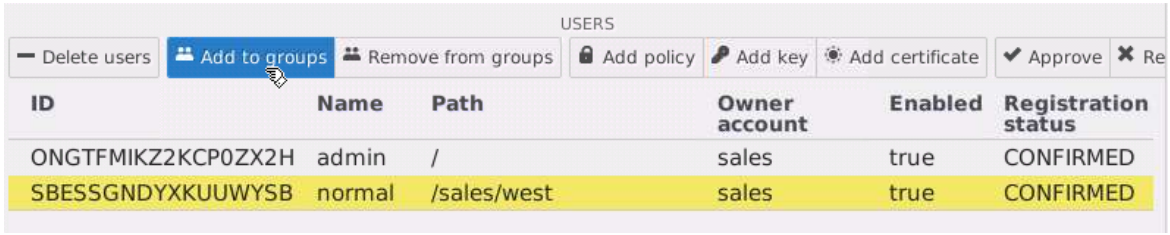


8. Click the user **normal** and then click the **Password** pencil icon in the PROPERTIES panel to create an initial login password for the new user.

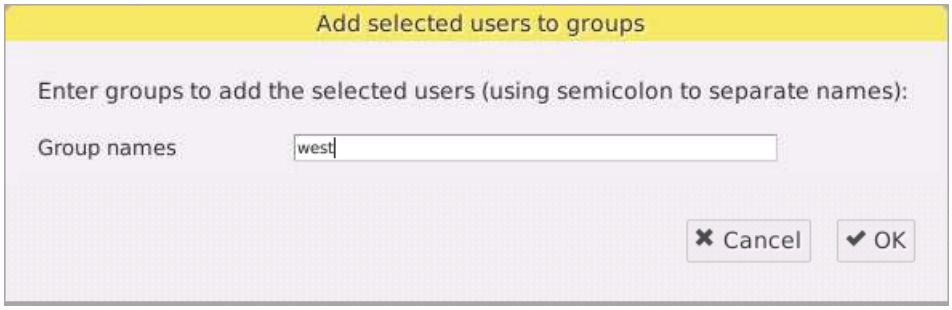
9. In the Change password window, enter the current *admin* users password (it should be `passwordNN`), and then enter `newuser` as the initial password for the new user. Click **OK**.



10. With the user **normal** in the list of users still selected, click the **Add to groups** button.

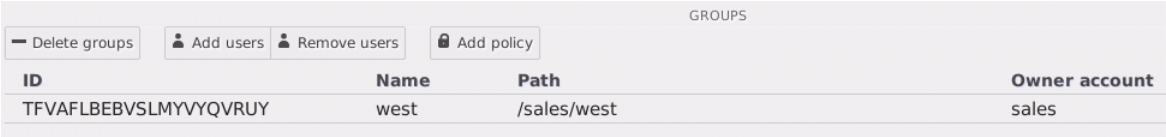


11. In the Add selected users to groups window, type `west` in **Group names** and click **OK**.



12. With the **normal** user still selected in the main panel, click **Membership groups** in the PROPERTIES panel.

13. Which group is the user *normal* a member of?



14. Sign out of the Administrator Console.

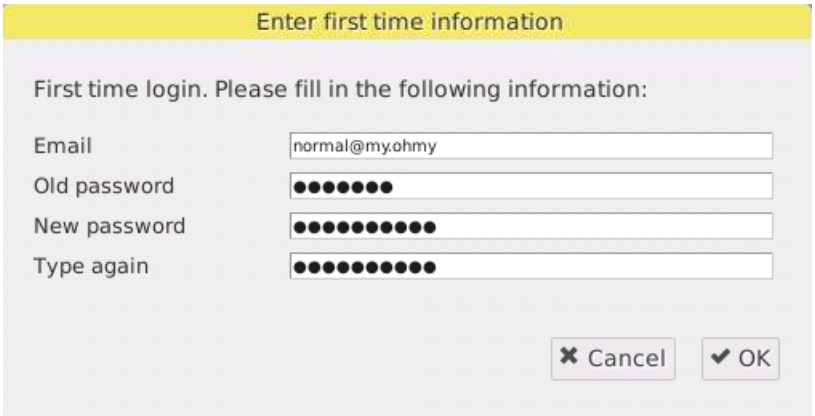
Log in as a non-administrative user

Desktop

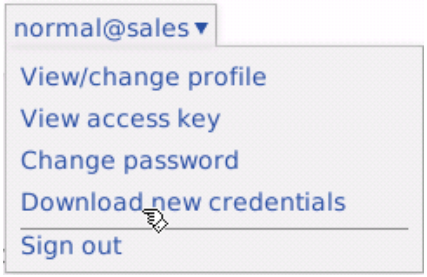
In this section of the lab you will log in to the Eucalyptus Administrator Console as a non-administrative user named *normal*. Once logged in, you will perform the first time log in steps and download the user's credentials to the Debian desktop.

1. Log in to the Administrator Console as the non-administrative user `normal` that you added in the previous section of the lab. The password should be `newuser`. Be sure to change the **Account** to the `sales` account.

2. Fill in the first-time login information. Enter an email address of your choice, enter the old password (it should be `newuser`), and as the new password enter the standard `passwordNN` password, where *NN* is the number of the student pod that you are assigned. Click **OK**.



3. Select **Download new credentials** on the login name drop-down menu to download the user's credentials. In the window that opens on the Debian desktop, select the **Save file** radio button and then click **OK**. Then close the Downloads window.



4. On the Debian desktop, open an xterm window and list the contents of the `/root/Downloads` directory. Do you see the zip file with the user's downloaded credentials? You will use this file in a later lab exercise. Close the xterm window when finished.

```
# ls /root/Downloads
```

5. Sign out of the Administrator Console.

Lab - Managing Eucalyptus Policies

In this lab exercise you will test a non-administrative user's default permissions in the cloud by performing operations using the Eucalyptus Administrator Console and euca2ools. You will then create a policy and assign it to the group of which the non-administrative user is a member. Finally, you will test the policy by logging in as the non-administrative user and attempting to perform tasks within the cloud. The non-administrative user is named *normal*.

Lab Objective:

- Test default permissions for a normal user using the Administrator Console
- Test default permissions for a normal user using the euca2ools
- Create a policy and assign it to a group
- Test the new policy's operation with a normal user

Test default permissions for a normal user using the Administrator Console

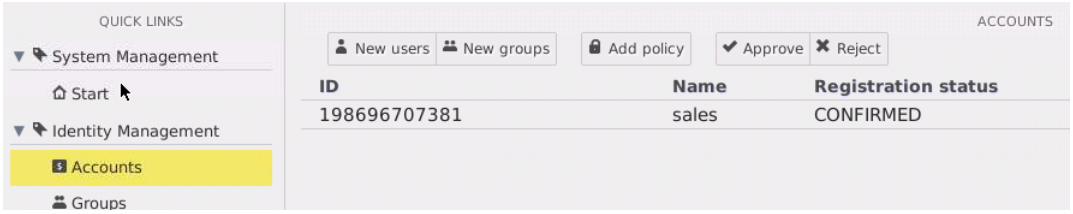
Desktop

In this section of the lab exercise you will log in to the Eucalyptus Administrator Console as the user *normal* in the *sales* account and test what you are able to do, and not able to do, using only the default user permissions.

1. From the Debian desktop, open the browser and use the URL https://<front_end_public_IP>:8443 to access the Administrator Console. Log in to the `sales` account as the user `normal` using the password `passwordNN`, where *NN* is the number of your student pod.

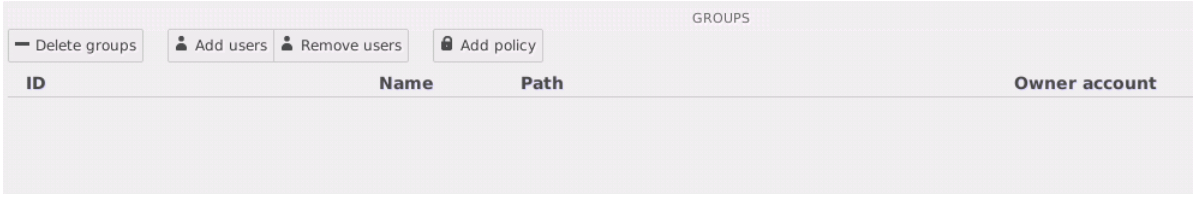
2. Note the current user name and account name above the main panel in the Administrator Console.

3. Click **Accounts** in the QUICK LINKS panel.



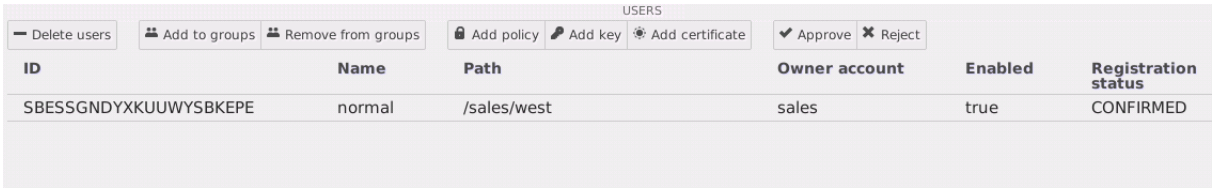
Which accounts is the user able to see? Is this all of the accounts or not?

4. Click **Groups** in the QUICK LINKS panel.



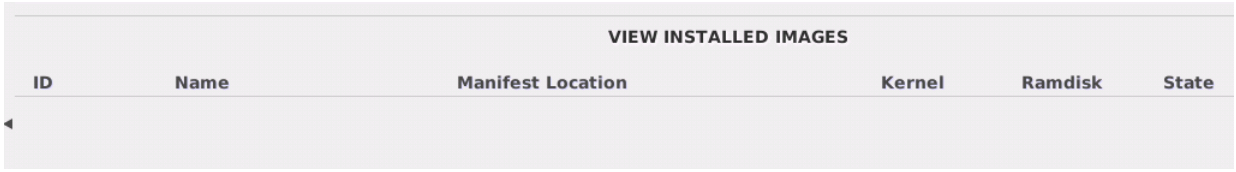
Which groups is the user able to see?

5. Click **Users** in the QUICK LINKS panel.



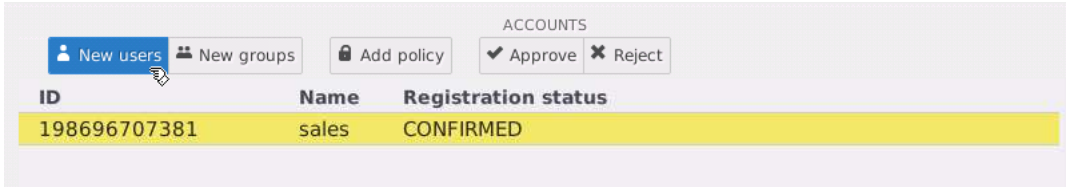
Which users is the user able to see?

6. Click **Images** in the QUICK LINKS panel.

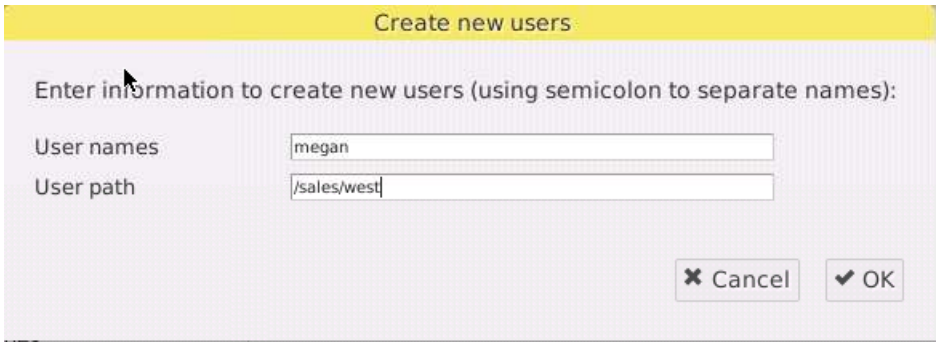


Which images is the user able to see?

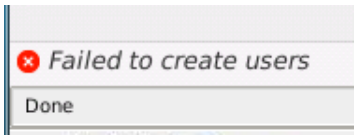
7. Click **Accounts** in the QUICK LINKS panel, click the **sales** account in the main panel, and then click the **New users** button.



8. In the Create new users window, create another new user in the *sales* account. Choose any user name and type `/sales/west` in the **User path**. Click **OK**.



9. Note the message at the bottom-left corner of the Administrator Console.



Was the non-administrative user allowed to add a new user account?

10. Sign out of the Administrator Console but do not close the browser window.

Test default permissions for a normal user using euca2ools

In this section of the lab you will use euca2ools to explore what you are able to do, and not able to do, using only the default user permissions. In order to test euca2ools as the user *normal*, you will need to create a new Linux account and add the user *normal*'s cloud credentials to this account.

1. Desktop On the Desktop, open an SSH session to the front-end host if an SSH session is not currently open.

```
# ssh <front_end_public_IP>
```

2. On the Debian desktop open an xterm window and use the Secure Copy command to copy the credentials (`.zip` file) for the user *normal* from the Debian desktop's `/root/Downloads` directory to the `/tmp` directory on the front-end host. The credentials were downloaded to a `.zip` file in an earlier lab exercise.) The IP address and password of your front-end host is listed on your student handout.

```
# scp /root/Downloads/euca2-normal-x509.zip <front_end_public_IP>:/tmp
```

3. Front End On the front-end host create a new Linux user account for the user *normal*. Assign the Linux account the password `passwordNN`, where *NN* is the number of the student pod that your were assigned.

```
# useradd -m normal
# passwd normal
# ls -a /home/normal
```

4. On the front-end host, switch to the new account for the user *normal*.

```
# su  -  normal
# whoami
# pwd
```

5. While working as the user *normal*, create a `/home/normal/.euca` directory and copy the credential zip file from `/tmp` to the `.euca` directory.

```
# mkdir .euca
# cd .euca
# cp /tmp/euca2-normal-x509.zip .
# ls
```

6. While working as the user *normal*, unzip the zip file and force the user's shell to read the `eucarc` file and set the EC2 environment variables.

```
# unzip euca2-normal-x509.zip
# ls
# source eucarc
# env | grep EC2
```

7. While working as the user *normal*, use the `euca-describe-images` command to list any existing images.

```
# euca-describe-images
```

Were there any images listed? There should not have been.

8. While working as the user *normal*, use the `euca-describe-groups` command to list any existing security groups.

```
# euca-describe-groups
```

Were there any security groups listed? There should not have been.

9. While working as the user *normal*, use euca-add-keypair to create a key pair.

```
# euca-add-keypair normal
```

Were you able to create a new key pair?

10. Return to working as the root user by exiting the Linux shell that is operating as the user *normal*.

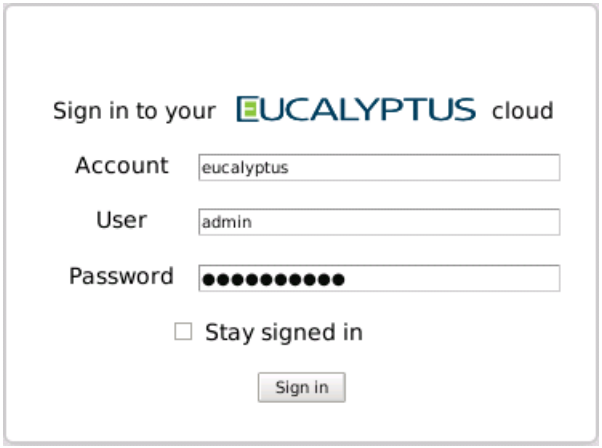
```
# exit
# whoami
```

Create a policy and assign it to a group

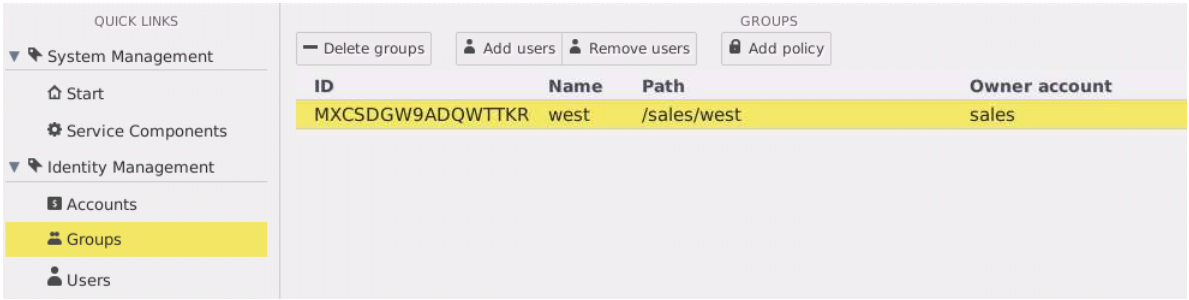
Desktop

In this section of the lab you will add a policy to the group *west* in the *sales* account. You will create the policy using the online Amazon Policy Generator tool and then copy and paste the policy text to the Eucalyptus Administrator Console. You will also view an existing policy in the Administrator Console.

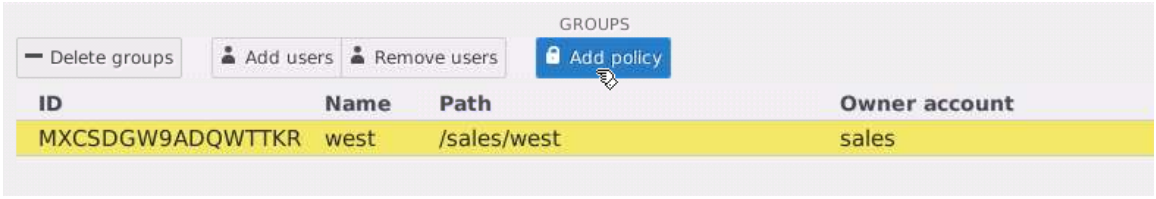
1. From the Debian desktop, open the browser and use the URL https://<front_end_public_IP>:8443 to access the Administrator Console. Log in to the `eucalyptus` account as the user `admin` using the password `passwordNN`, where *NN* is the number of your student pod.



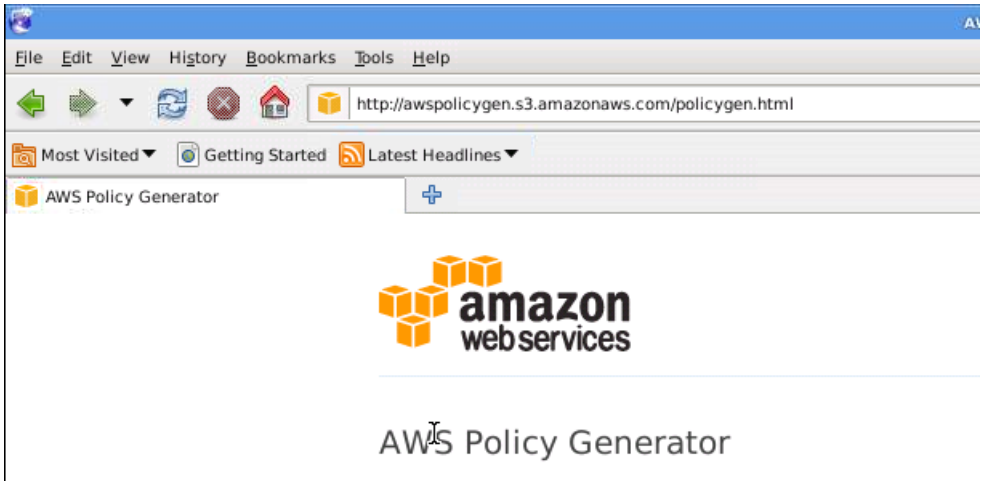
2. Click **Groups** in the QUICK LINKS panel and then select the *west* group in the main panel.



3. Click the **Add policy** button to add a policy to the *west* group.



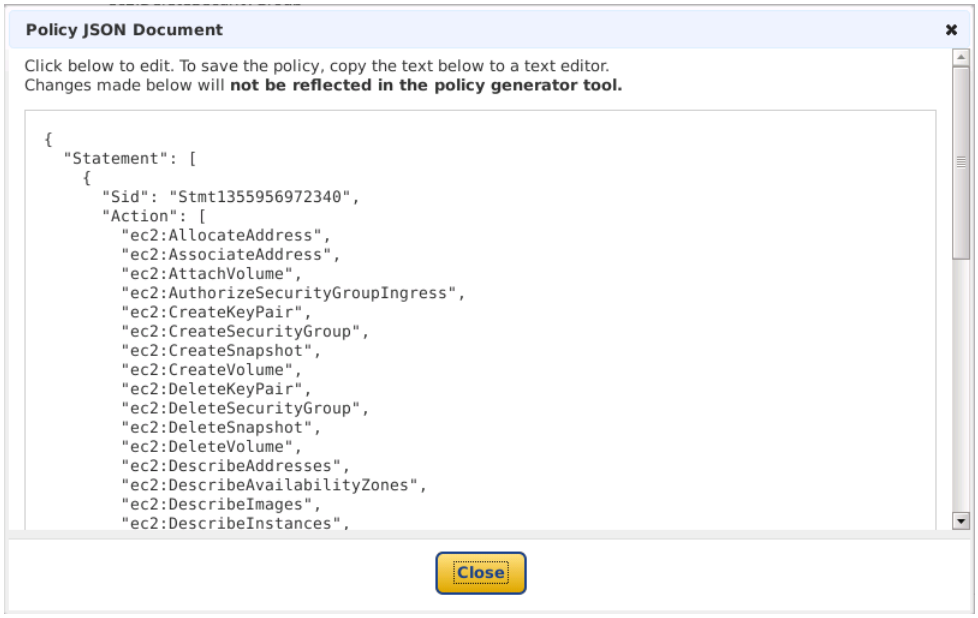
4. With the Add new policy window opens in the Administrator Console, open a new browser on the Debian desktop. Type the URL *http://awspolicygen.s3.amazonaws.com/policygen.html* in the new browser window to open the online AWS Policy Generator tool.



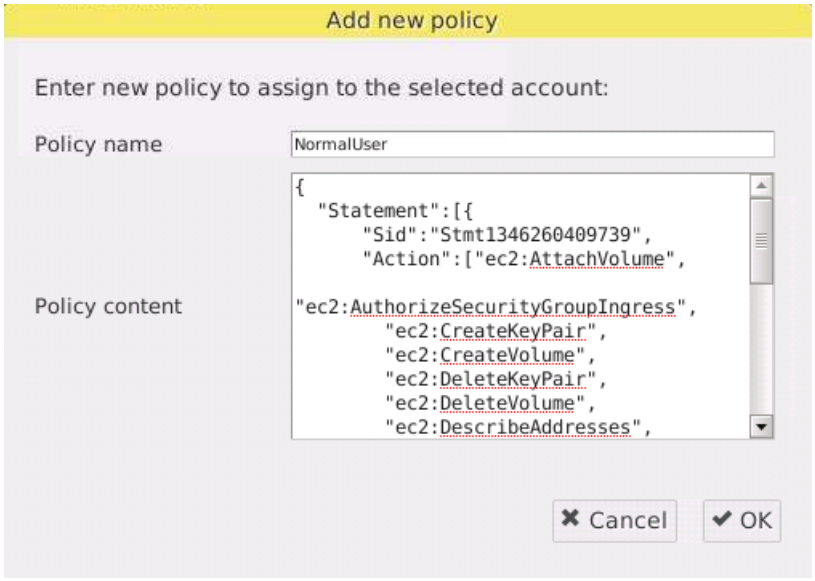
5. In the Policy Generator tool, set **Select Type of Policy** to **IAM Policy**. Leave **Effect** set to **Allow**. Set **AWS Service** to **Amazon EC2**. Under **Actions**, select the 28 check boxes for; **AllocateAddress**, **AssociateAddress**, **AttachVolume**, **AuthorizeSecurityGroupIngress**, **CreateKeyPair**, **CreateSecurityGroup**, **CreateSnapshot**, **CreateVolume**, **DeleteKeyPair**, **DeleteSecurityGroup**, **DeleteSnapshot**, **DeleteVolume**, **DescribeAddresses**, **DescribeAvailabilityZones**, **DescribeImages**, **DescribeInstances**, **DescribeKeyPairs**, **DescribeSecurityGroups**, **DescribeSnapshots**, **DescribeVolumes**, **DetachVolume**, **DisassociateAddress**, **GetConsoleOutput**, **RebootInstances**, **ReleaseAddress**, **RevokeSecurityGroupIngress**, **RunInstances**, and **TerminateInstances**. When finished making these selections click **Add Statement**.

Effect	Action	Resource	Conditions
Allow	• ec2:AllocateAddress • ec2:AssociateAddress • ec2:AttachVolume • ec2:AuthorizeSecurityGroupIngress • ec2:CreateKeyPair • ec2:CreateSecurityGroup • ec2:CreateSnapshot • ec2:CreateVolume • ec2:DeleteKeyPair • ec2:DeleteSecurityGroup • ec2:DeleteSnapshot • ec2:DeleteVolume • ec2:DescribeAddresses • ec2:DescribeAvailabilityZones • ec2:DescribeImages • ec2:DescribeInstances • ec2:DescribeKeyPairs • ec2:DescribeSecurityGroups • ec2:DescribeSnapshots • ec2:DescribeVolumes • ec2:DetachVolume • ec2:DisassociateAddress • ec2:GetConsoleOutput • ec2:RebootInstances • ec2:ReleaseAddress • ec2:RevokeSecurityGroupIngress • ec2:RunInstances • ec2:TerminateInstances	*	*None*

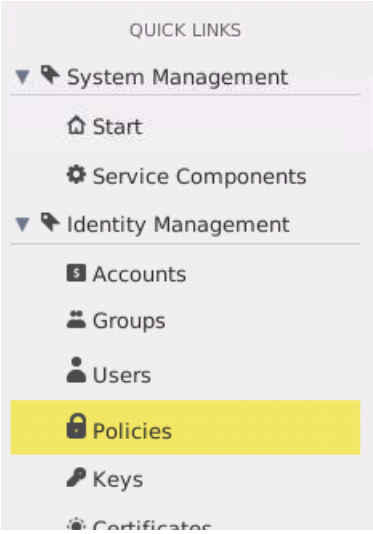
6. Click **Generate Policy**. Do not click **Close** yet.



7. Copy the policy to your clipboard by selecting the policy text and pressing Ctrl-C. Be sure to capture any beginning and ending curly braces or square brackets in the policy text.
8. Switch to the Administrator Console and use Ctrl-V to paste the policy into the **Policy Content** text box. Name the policy `NormalUser` (no whitespace) in the **Policy name** text box. Click **OK**.



9. To view a policy, click **Policies** in the QUICK LINKS panel.



10. Click the **NormalUsers** policy in the main panel and note that the policy text appears in the PROPERTIES window. The policy text can be copied and pasted to create additional policies for other users and groups.

11. Log out of the Administration Console, and then close both the browser. Also close the browser with the Amazon policy generator utility.

Test the policy's operation with a normal user

In this section of the lab you will test the new policy by working as the user *normal* using euca2ools. The user *normal* is part of the group *west* which was assigned the policy *NormalUser* in the previous section of this lab exercise. As the user *normal*, you will perform the tasks associated with normal daily cloud use such as describing images, addresses, and security groups. The user *normal* could not perform any of these operations before a policy was put into effect.

Note: In a later lab you will also test the policy by creating a keypair, modifying a security group, viewing images, launching and terminating an instance, manage an elastic IP address, and manage volumes and snapshots, all using the Eucalyptus User Console.

1. Desktop On the Desktop, open an SSH session to the front-end host if an SSH session is not currently open.

```
# ssh <front_end_public_IP>
```

2. On the front-end host, switch to the new account for the user *normal*.

```
# su  -  normal
# whoami
```

3. While working as the user *normal*, force the user's shell to read the `eucarc` file and set the EC2 environment variables.

```
# source .euca/eucarc
# env | grep EC2
```

4. While working as the user *normal*, use the `euca-describe-images` command to list any existing images.

```
# euca-describe-images
```

   Were there any images listed?

5. While working as the user *normal*, use the `euca-describe-groups` command to list any existing security groups.

```
# euca-describe-groups
```

   Were there any security groups listed?

6. Notice that the commands have all run without error and have produced output. This indicates that the cloud is working correctly and the user has sufficient permissions to view the cloud resources.
7. Return to working as the root user by exiting the Linux shell that is operating as the user *normal*.

```
# exit
# whoami
```

Eucalyptus User Console

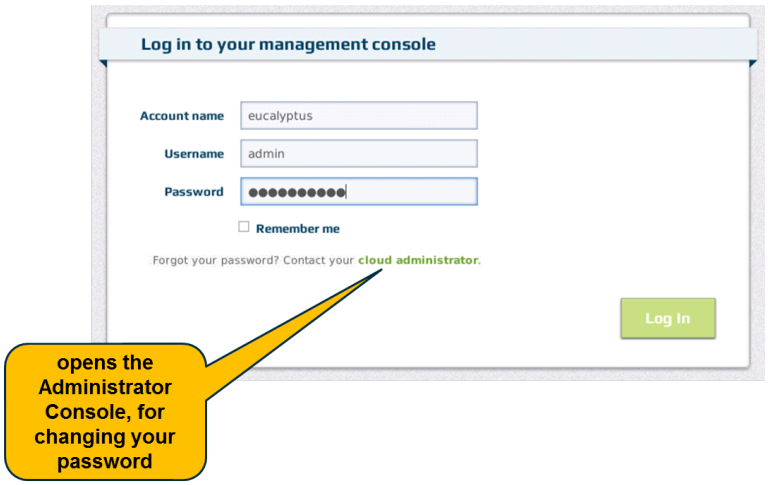
The Eucalyptus User Console is an easy-to-use Web-based user interface. It can be used to manage images, instances, key pairs, IP addresses, volumes, snapshots, and security groups. It is an alternative to the command-line euca2ools.

It is installed as a service that is access by a URL. The format of the URL is https://<host>:<port>. The <host> is the IP address or DNS name of the machine where the service was installed. The <port> is determined by the cloud administrator when they configure the service. The default port is 8888. The following illustrates access to the User Console.

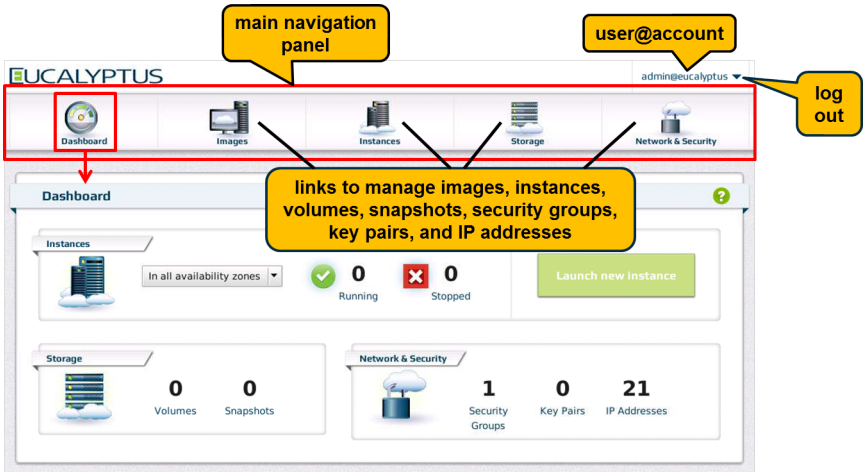
Users will need an account name, login name, and password in order to log in to the User Console.

User Console log in

To log in to the User Console, a user will need an account name, user name, and password. This can be created by the cloud administrator using the Eucalyptus Administrator Console. Creating new accounts and users is discussed in another section of the course.



User Console Dashboard



The Dashboard is the initial screen you are presented with when you log in to the User Console. The Dashboard quickly provides information about the status of instances, storage, and network and security resources. You can also use the Dashboard to navigate to other screens where you can perform actions in the cloud. Performing actions in the cloud using the User Console is discussed in other sections of the course when they are relevant the topic.

Installing the User Console

You may install the User Console service on any host that has access to the Cloud Controller and the cloud-user population. Installation on the Cloud Controller host is allowed if users have access to the host and there are sufficient resources to maintain acceptable performance.

There are two software requirements for the User Console service. You must install the Eucalyptus release RPM file and the Extended Packages for Enterprise Linux (EPEL) RPM file. These RPM files install the `eucalyptus-release.repo` and `epel.repo` files in the `/etc/yum.repos.d` directory.

```
# yum install http://downloads.eucalyptus.com/software/eucalyptus/3.2/ \
        centos/6/x86_64/eucalyptus-release-3.2.noarch.rpm
# yum install http://downloads.eucalyptus.com/software/eucalyptus/3.2/ \
        centos/6/x86_64/epel-release-6.noarch.rpm
```

Once you have installed the prerequisites, install the User Console software using the following command:

```
# yum install eucalyptus-console
```

Configuring the User Console

Once the User Console software is installed, it must be configured for operation within your cloud. The main configuration file for the User Console is `/etc/eucalyptus-console/console.ini`. It is composed of several sections and individual entries in the format of <parameter>: <value>.

There are three main parameters in the `[server]` section that must be configured.

The IP address of the Cloud Controller must be configured in the `clchost:` parameter. For example, if the IP address of your Cloud Controller is 172.16.162.1 then the entry would look like `clchost: 192.16.162.1`.

The port number where the User Console can be reached must be configured in the `uiport:` parameter. For example, if you want your User Console to respond to requests at port 9090, then the entry would look like `uiport: 9090`. The default port number is 8888.

You should also configure the URL of the Eucalyptus Administrator Console in the `support.url:` parameter. A user is not able to change their password in the User Console. However, the User Console login window contains a link that can take the user to the Eucalyptus Administrator Console where they can change their password. This link works in your cloud by adding the Eucalyptus Administrator Console URL to the `support.url:` parameter. For example, if your Eucalyptus Administrator Console is reachable at https://192.16.162.1:8443, then the entry would look like `support.url: https://192.16.162.1:8443`.

There are other configuration parameters in this file that allow you to configure custom SSL keys, your locale, session timeouts, polling frequencies, and others settings. For more information about the configuration file, see the *Eucalyptus User Console Guide* at *http://www.eucalyptus.com/docs*.

Configuring vmtypes

In Eucalyptus 3.2, the User Console must be manually configured to match the vmtypes configuration of your cloud. This is only necessary if you have customized the vmtypes values in your cloud. If you are using the default values in your cloud, then the default values in the User Console will match.

Note: Automatic configuration is planned for the Eucalyptus 3.3 release.

The vmtypes values are configured in `[instance_type]` section of the `/etc/eucalyptus-console/console.ini` file.

The default values are shown below:

```
[instance_type]
m1.small.cpu: 1
m1.small.mem: 128
m1.small.disk: 2
c1.medium.cpu: 2
c1.medium.mem: 128
c1.medium.disk: 5
m1.large.cpu: 2
m1.large.mem: 512
m1.large.disk: 10
m1.xlarge.cpu: 2
m1.xlarge.mem: 1024
m1.xlarge.disk: 10
c1.xlarge.cpu: 4
c1.xlarge.mem: 2048
c1.xlarge.disk: 10
```

How to change the vmtypes in your cloud is covered in another training lesson.

Starting the User Console

Once the `/etc/eucalyptus-console/console.ini` file has been configured, you will need to start the User Console service using the following command:

```
# service eucalyptus-console start
```

If changes are made to the `console.ini` file, then restart the User Console service.

```
# service eucalyptus-console restart
```

You might need to refresh your browser window too.

Managing Key Pairs

The Eucalptus User Console is used to manage key pairs. You can create, list, and delete key pairs.

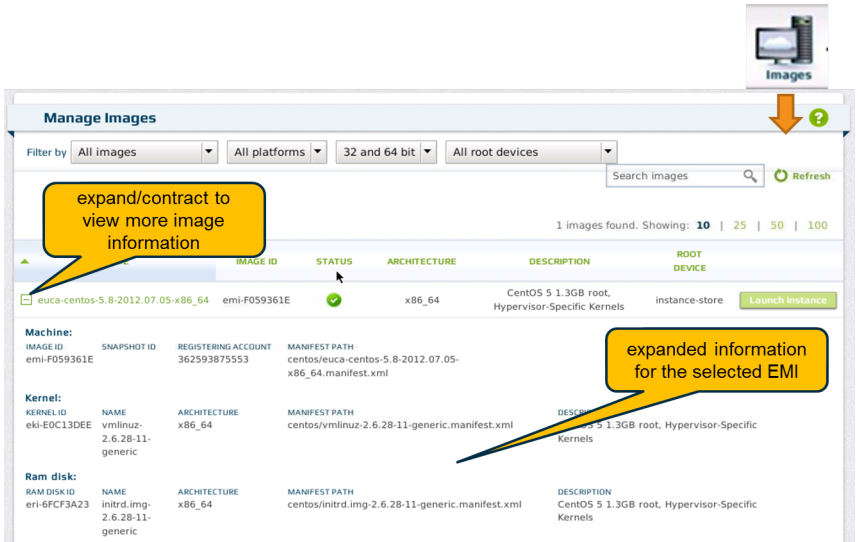
Manage key pairs in the User Console by clicking the **Network & Security** icon and then selecting **Key Pairs** from the drop-down menu.

In the screen capture above you can see a key pair named *adminkey*. You can click the check box next to the key which activates the **More actions** button. This button allows you to delete the selected key pair.

There are other buttons that allow you to create a new key pair, or import an existing SSH key pair.

Listing Images

You may also list images registered with the Cloud Controller using the Eucalyptus User Console.

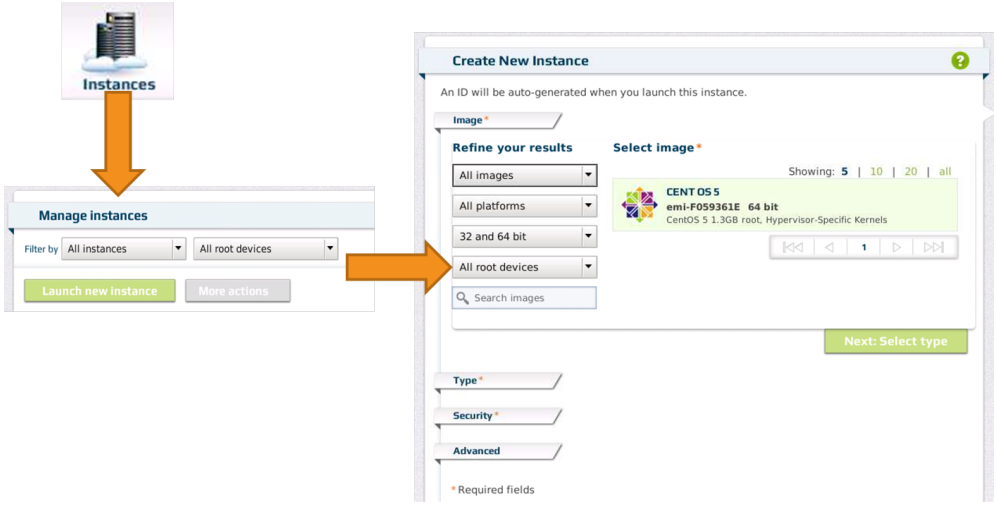


In normal view, only the EMIs are shown. However, by clicking the plus (+) icon next to the EMI, it will expand to display more information. The expanded information view is shown above.

Starting an Instance

It is also possible to launch new instances using the Eucalyptus User Console. There is more than a single method to launch in instance using the User Console. This section will describe one of the methods.

Launch an instance in the User Console by clicking the **Instances** icon and then clicking the **Launch new instance** button.



In the Create New Instance window you make a series of choices in order to launch an instance. The choices are listed below:

- **Image** section - Select your EMI image. Several filters exist to find the exact EMI you want. Filters include:
  - All images
  - Images owned by me
  - All platforms
  - Linux
  - Windows
  - 32 and 64 bit
  - 32 bit
  - 64 bit
  - All root devices
  - Instance-store root device
  - EBS root device
- **Type** section
  - Select one of the five vmtypes
  - Select the number of instances to launch
  - Select a specific availability zone or let the cloud pick the availability zone
- **Security** section
  - Select a key pair
  - Select a security group

At this point you are offered the choice to launch the instance or continue on to the **Advanced** section where you can make additional choices.

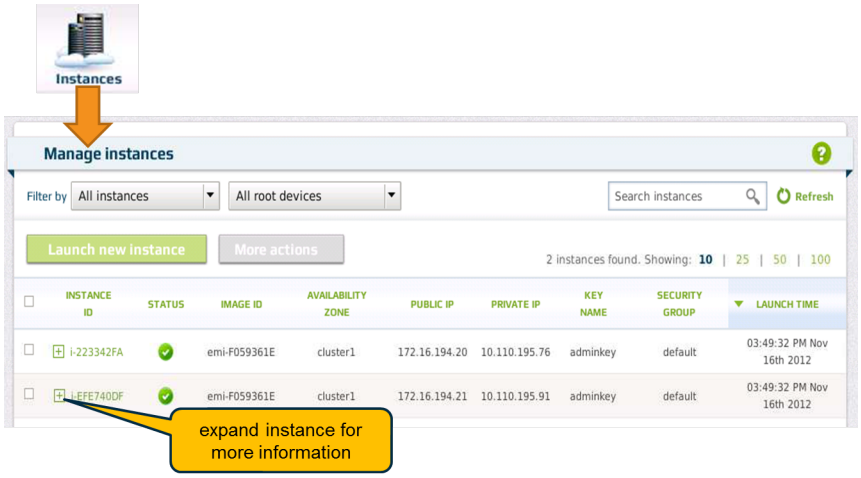
- **Advanced** section
  - Manually enter user data that will be made available to the instance when it boots
  - Browse to a file that contains user data that will be made available to the instance when it boots
  - Choose a specific EKI with which to launch the EMI (overrides any default image)
  - Choose a specific ERI with which to launch the EMI (overrides any default image)
  - Choose to launch the instance with only a private IP address and no public IP address

At this point you may launch the instance.

If at any time you want to go back and change your selections before you launch the instance, you may by clicking the appropriate section title. Clicking the section title reopens the section and allows you to change your selections.

Listing Instances

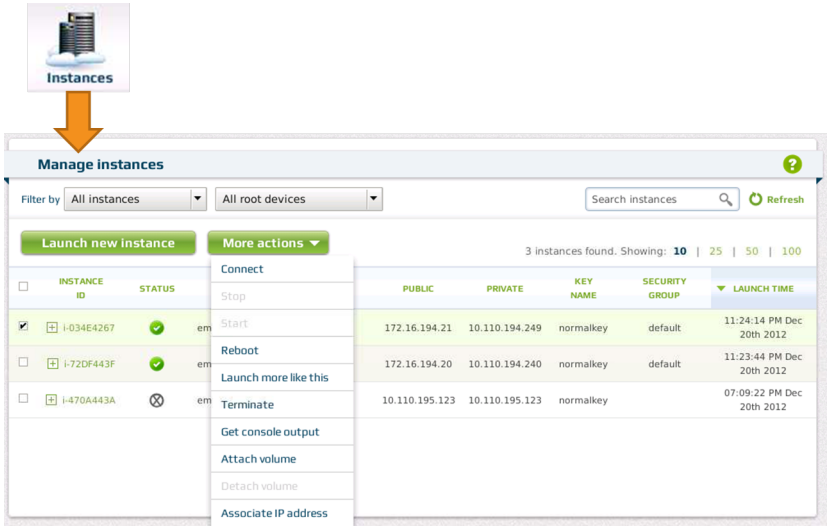
You can list instances using the Eucalyptus User Console.



For more information about the instance, including the EKI and ERI used to launch it, click the plus (+) sign to the left of the instance ID.

Controlling an Instance

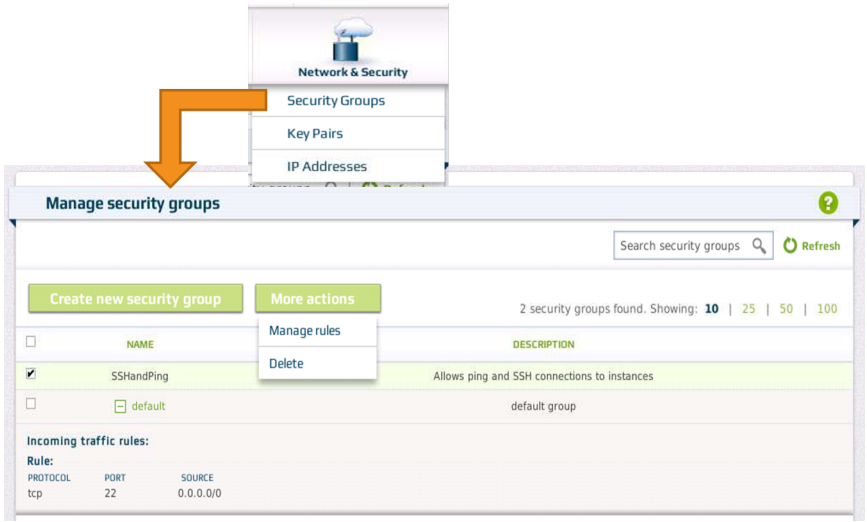
An instance store or EBS-backed instance can be rebooted or terminated. Data stored in an ephemeral instance is not lost during a reboot, but is lost when an instance is terminated.



Note: Some versions of hypervisors might not reconnect a volume to the instance after a reboot and manual intervention might be necessary.

Managing Security Groups

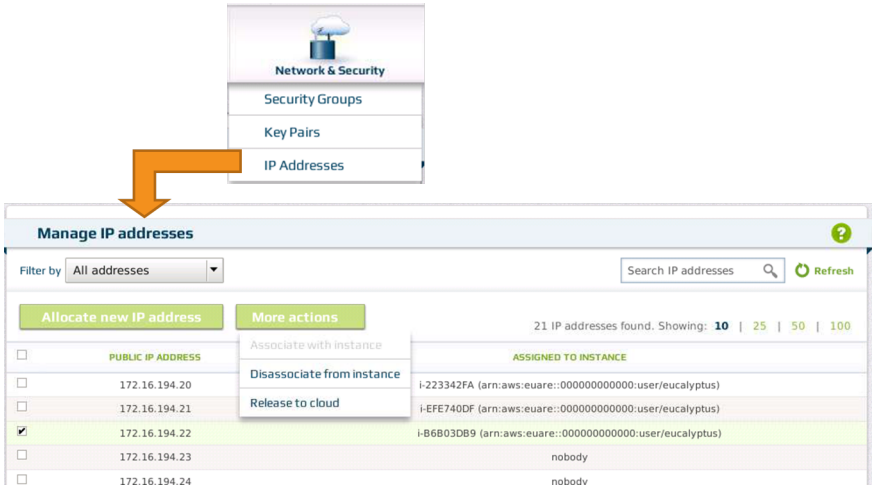
The Eucalyptus User Console also allows users to manage their security groups. Users can list their security groups, add or delete a security group, and authorize or de-authorize outside connection access. To view or manage security groups in the User Console, click **Network & Security** and then select **Security Groups**.



Clicking the **Create new security group** button opens a window that allows you to name a group and then add one or more rules to it. The security group *SSHandPing* was added by a user. Notice that it is selected, which allows the user to click the **More actions** button and choose to either manage the security group's rules or delete the security group.

Managing Elastic IP Addresses

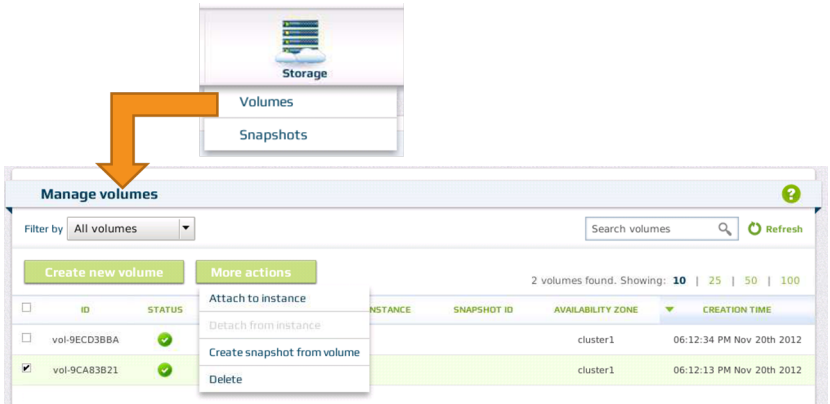
The Eucalyptus User Console is used to manage elastic IP addresses in the same way that euca2ools are used. To view or manage elastic IP addresses using the User Console, click **Network & Security** and then select **IP Addresses**.



To reserve a public IP address, click the **Allocate new IP address** button. To associate a reserved address with an instance or dissassociate an address from an instance, click the appropriate choice beneath the **More actions** button. The **More actions** button also allows you to release a reserved address back into the pool of available public IP addresses.

Managing Volumes

You can manage volumes using the Eucalyptus User Console. To manage volumes click **Storage** in the User Console and then select **Volumes** from the menu.

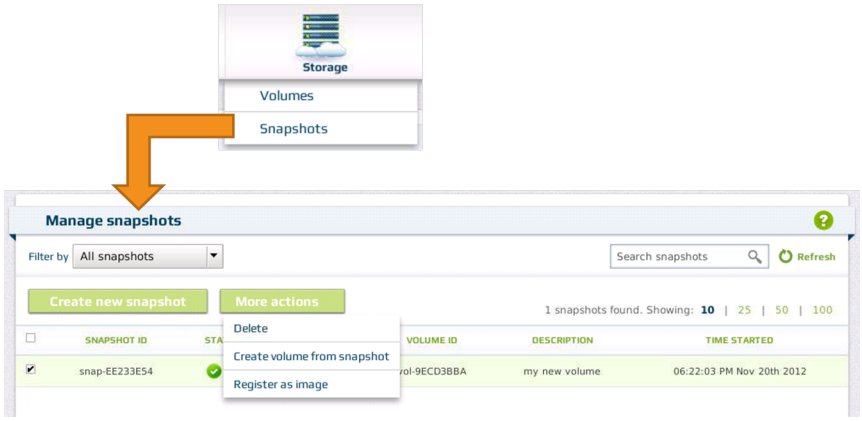


To create a volume click the **Create new volume** button. To attach or detach a volume from an instance, select the check box next to the volume and click the **More actions** button. Then select either **Attach** to instance or **Detach** from instance, as appropriate. You may even delete a selected volume by selecting **Delete** beneath the **More actions**

button. To create a snapshot of a selected volume, select **Create snapshot from volume** beneath the **More actions** button.

Managing Snapshots

The Eucalyptus User Console provides the same snapshot functionality that euca2ools provides. To access snapshot management in the User Console, click the **Storage** button and then select **Snapshots** from the drop-down menu.



To create a new snapshot of a volume, click the **Create new snapshot** button and follow the on-screen directions. To delete a snapshot, create a volume from a snapshot, or register a snapshot as a boot-from-EBS image, click the check box next to a snapshot and then select the appropriate choice beneath the **More actions** button.

Note: Creating boot-from-EBS images is explained later in this lesson.

Lab - Installing, Configuring, and Using the Eucalyptus User Console

In this lab exercise you will install, configure, and use the Eucalyptus User Console. Using the User Console you will create a new key pair. Then you will launch, connect to, and terminate and instance. You will also manage a security group, elastic IP addresses, volumes, and snapshots.

Lab Objective:

- Install and configure the Eucalyptus User Console
- Log in to the User Console
- Create a key pair
- Launch an instance
- Connect to a running instance
- Terminate an instance
- Create a security group
- Modify a security group
- Delete a security group
- Reserve an elastic IP address
- Assign an elastic IP address
- Unassign an elastic IP address
- Create a storage volume
- Attach a volume to an instance

- Detach a volume from an instance
- Snapshot a volume
- Create a volume from a snapshot
- Delete a volume and snapshot

Install and Configure the Eucalyptus User Console

In this section of the lab you will install and configure the Eucalyptus User Console. Once you have completed this task, you will start the User Console service.

1. Desktop If necessary, from the Debian desktop open an SSH session to the front-end host.

```
# ssh <front_end_public_IP>
```

2. Front End One the front-end host, use the `yum` command to install the User Console service.

```
# yum install eucalyptus-console
```

Note: The User Console installation depends on the `eucalyptus-release.repo` and `epel.repo` files in the `/etc/yum.repos.d` directory. These files have already been installed on the front-end host as part of the initial Eucalyptus installation.

3. Front End Open the `/etc/eucalyptus-console/console.ini` file using either the `vi` or `nano -w` editor. Leave the file open for editing.

```
# vi /etc/eucalyptus-console/console.ini
```

4. Front End While in the editor, make the following changes in the `console.ini` file. All the other default settings will work in the lab environment. Save your changes and close the editor when you are finished.

```
clchost: <front_end_public_IP>
support.url: https://<front_end_public_IP>:8443
```

5. Front End On the front-end host, start the User Console service.

```
# service eucalyptus-console start
```

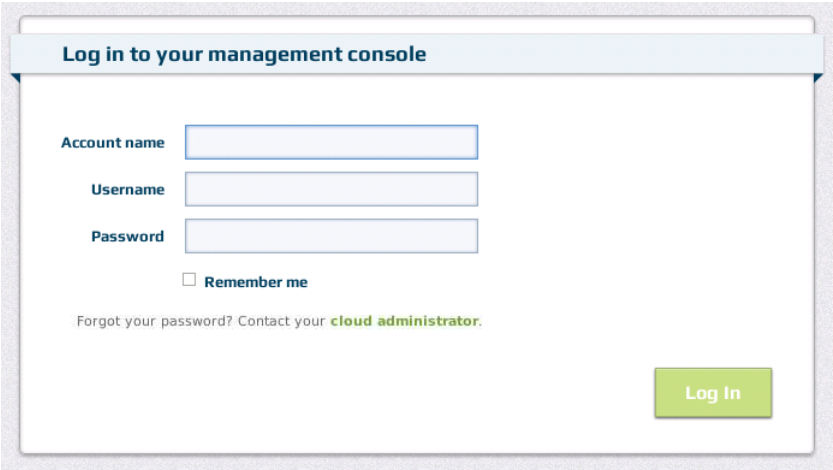
Log In to the User Console

Desktop

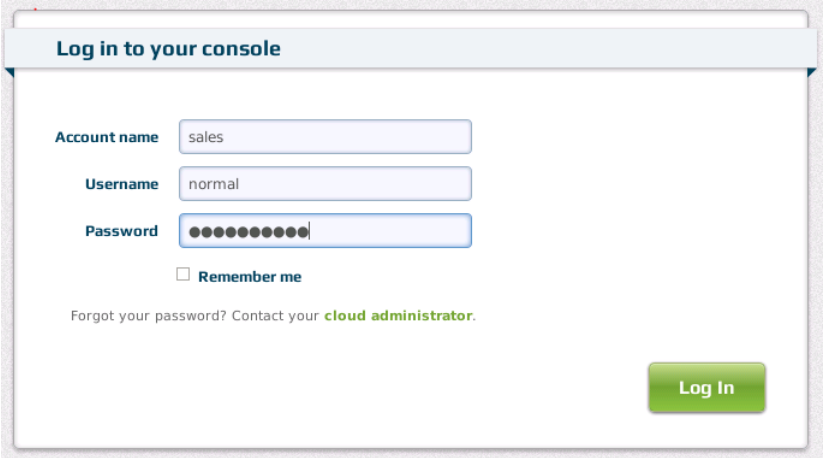
The purpose of this section of the lab is only to learn how to navigate in the User Console interface and to see what types of actions and information are available. Later lab exercises will teach you to use specific User Console functionality to create key pairs, launch instances, and manage storage, network, and security resources.

With the User Console service installed, configured, and running, you will open a browser and log in to the User Console. Once in the User Console you will explore the interface *without* making changes. Later labs will explain User Console operations and provide hands-on experience using the interface.

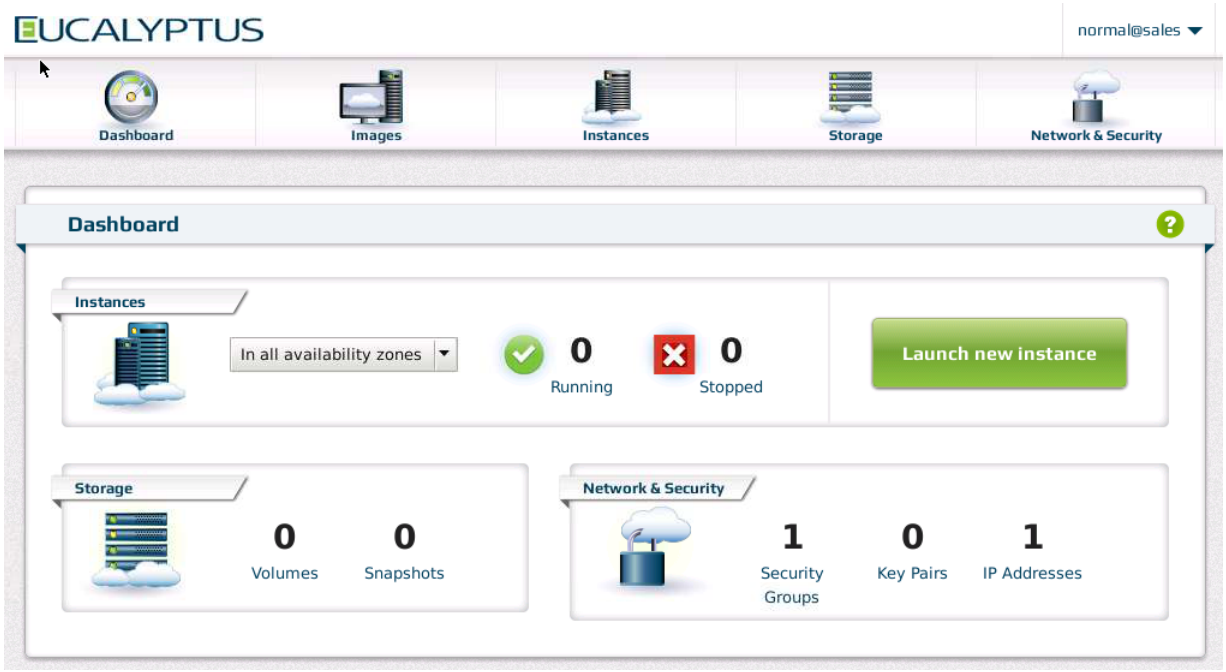
1. On the Debian desktop, open the browser and connect to the User Console using the URL https://<front_end_public_IP:8888. Because the User Console server uses a self-signed certificate, you will need to confirm the security of the connection to the browser.



2. At the login window, sign in to the cloud as the user `normal` in the `sales` account using the password of `passwordNN`, where *NN* is the number of your student pod. Then click **Log In**.



The Dashboard window will appear on your screen.



3. View the types of information available to you in the Dashboard window of the User Console and answer the following questions. Do not make changes at this time.

   How many `Running` and `Stopped` instances do you have in your cloud?

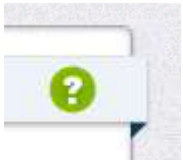
   How many `Volumes` and `Snapshots` do you have in your cloud?

   How many `Security Groups` do you have in your cloud?

   How many `Key Pairs` do you have in your cloud?

   How my public `IP Addresses` do you have in your cloud?

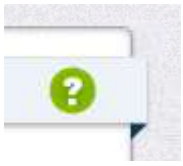
4. In the Dashboard window of the User Console, click the help (?) icon in the top-right corner.



   The help window for the Dashboard opens. The help icons in each window are context-sensitive and display help information relative to the current window.

5. In the help window, click the icon that will open a separate help window and return you to the Dashboard window.

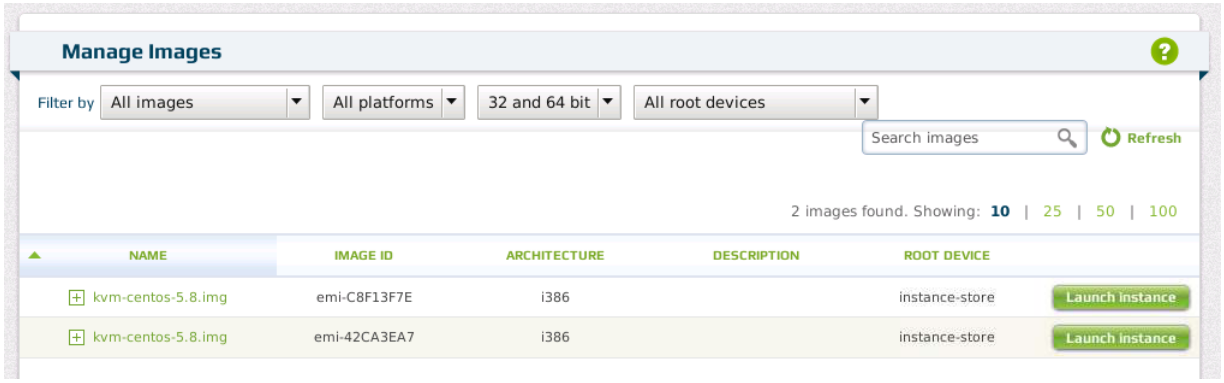
6. Close the separate help window.
7. In the Dashboard window of the User Console, click the help (?) icon in the top-right corner again.



8. In the help window, click the icon that will return you to the Dashboard window.

9. In the User Console, click the **Images** icon.

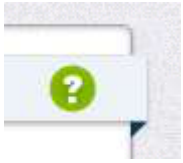
The Manage Images window opens. How many images are in your cloud?



Note: If you see no images in the list, this is likely due to a new Eucalyptus behavior. In prior versions of the software, any images uploaded by the cloud administrator were visible to all cloud users by default. In the latest version, this is no longer the case. To fix this, log into the front-end server and run the following command for you EMIs, EKIs, and ERIs:

```
euca-modify-image-attribute -l -a all <EMI-XXXXXX>
```

10. In the Manage Images window, click the help icon in the top-right corner.

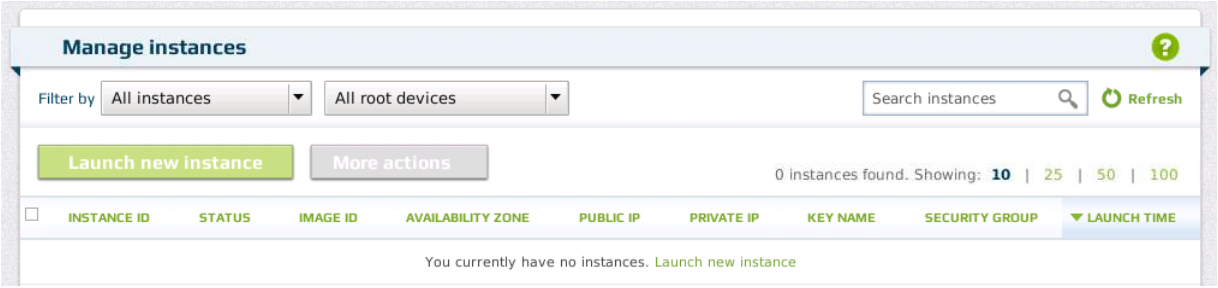


Is the help window that opens different from the Dashboard help window that you opened earlier?

11. Click the return icon to return to the Manage Images window.

12. In the User Console, click the **Instances** icon.

The Manage instances window opens. How many instances are in your cloud?



13. In the User Console, click once on the **Storage** icon.

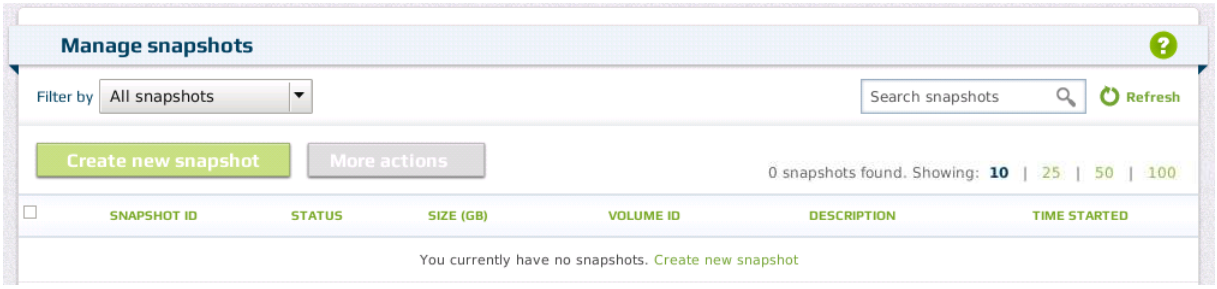
What drop-down menu choices appear on your screen?

14. On the **Storage** drop-down menu, click **Volumes**.

The Manage volumes window opens. How many volumes do you have?

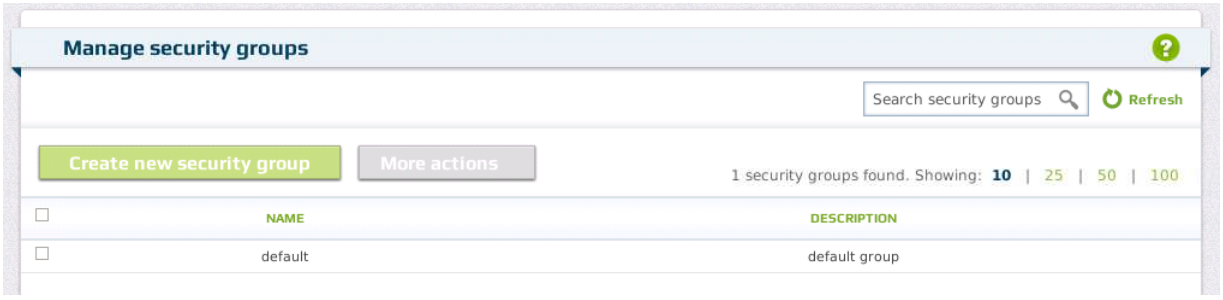
15. On the **Storage** drop-down menu, click **Snapshots**.

The Manage snapshots window opens. How many snapshots do you have?



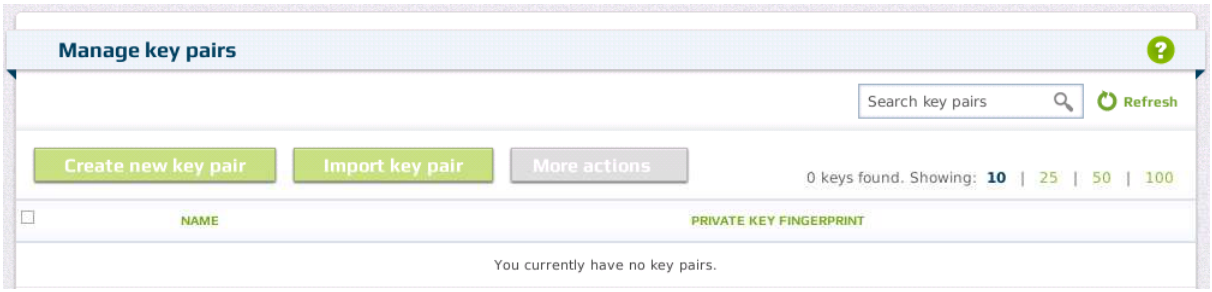
16. On the **Network & Security** drop-down menu, click **Security Groups**.

The Manage security groups window opens. How many security groups do you have? Do they have names?



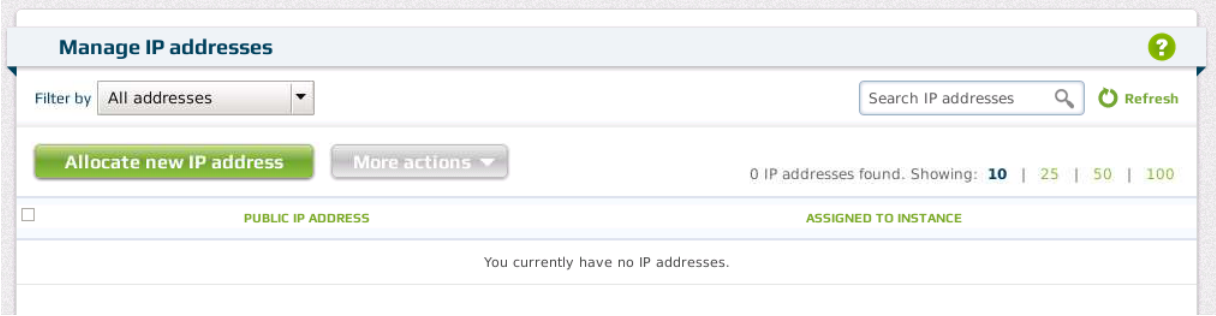
17. On the **Network & Security** drop-down menu, click **Key Pairs**.

The Manage key pairs window opens. How many key pairs do you have?



18. On the **Network & Security** drop-down menu, click **IP Addresses.**

The Manage IP addresses window opens. How many public IP addresses have been allocated (reserved) by the user *normal* in the cloud?



19. Remain logged in to the User Console.

Create a Key Pair Using the Eucalyptus User Console

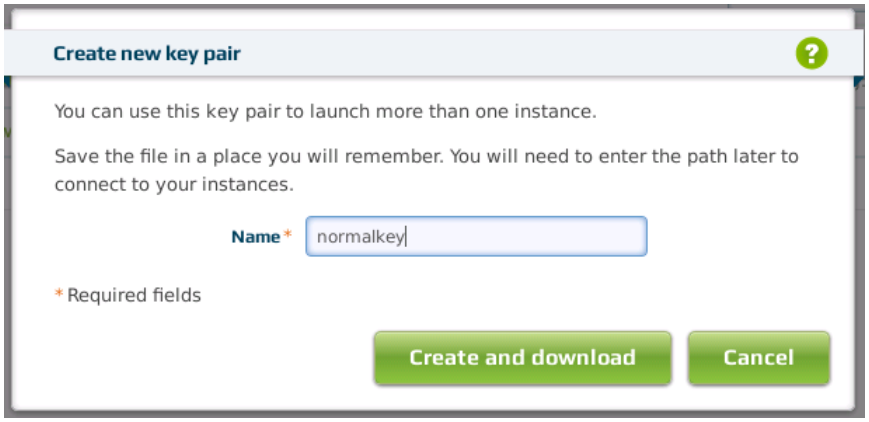
Before a user can launch a new instance, they must create a public/private key pair to use to authenticate to the instance during log in. The public key will automatically be added to the newly created instance. The private key must be used with Secure Shell (SSH) to connect to the instance. The Eucalyptus User Console provides the user a mechanism to create a key pair and to download the private key to a locally stored file.

1. Desktop On the Debian desktop, ensure that you are logged in to the User Console as the user *normal* in the *sales* account using the password of `passwordNN`, where *NN* is the number of your student pod.

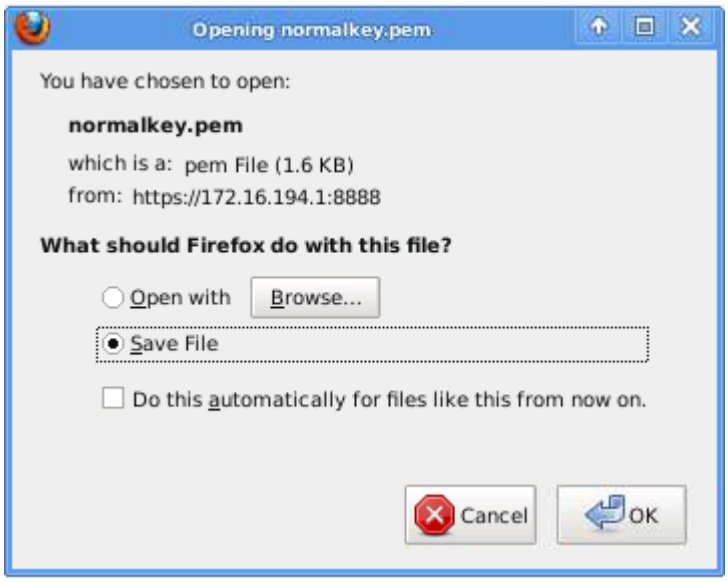
2. On the **Network & Security** drop-down menu, click **Key Pairs**.

3. Click the **Create new key pair** button.

4. Type `normalkey` in the Create new key pair window. Then click the **Create and download** button. This will create a new key pair for the *normal* user in the *sales* account.



5. Click **OK** to save the private key file of the *normal* user to a file on the Debian desktop system.



6. Close the Downloads window when it opens on your screen. The new key pair should be visible in the User Console window.

7. On the Debian desktop, open a new xterm window by clicking the terminal icon in the task bar.

8. In the open xterm window, list the files in the `/root/Downloads` directory.

```
# ls /root/Downloads
```

   You should see the file `normalkey.pem`. This is the private key for the *normal* user in PEM format.

9. In the open xterm window, change the permissions on the `normalkey.pem` file so that the security checks within the SSH program will allow it to be used when logging in to your instances.

```
chmod 600 /root/Downloads/normalkey.pem
```

10. In the open xterm window, use the Secure Copy program to copy the private key file on the Debian desktop to root's home directory on the front-end host. Use the public IP address of the front-end host. The public IP is listed in your student handout. When prompted for the password, the password should be `passwordNN`, where *NN* is the number of your student pod.

```
# scp /root/Downloads/normalkey.pem <front_end_public_IP>:/root
```

11. To prepare for future lab exercises, on the Debian Desktop copy the private key file from the `/root/Downloads` directory to the `/root` directory.

```
# cp /root/Downloads/normalkey.pm /root
```

12. Close the open xterm window on the Debian desktop.

```
# exit
```

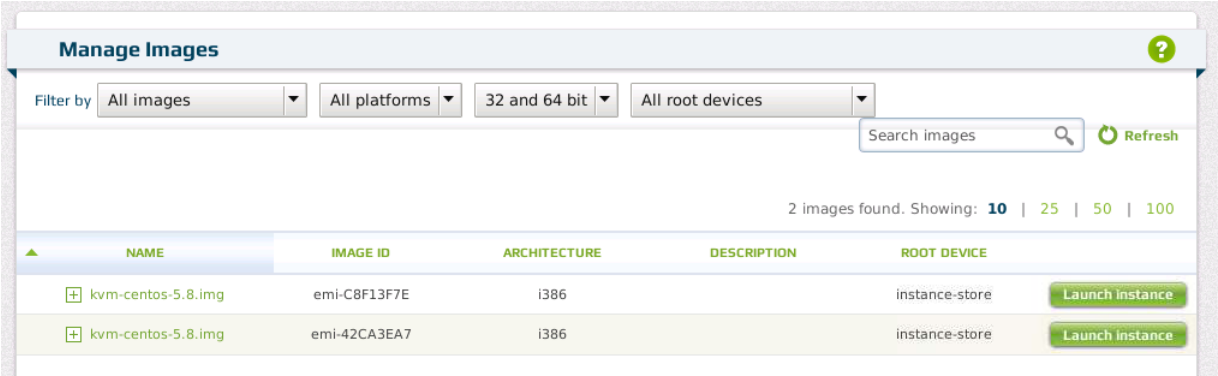
Launch an Instance Using the User Console

Desktop

In this section of the lab you will use the Eucalyptus User Console to launch a new instance using a key pair and an EMI image.

1. In the User Console, click the **Images** icon to return to the Manage Images window.

   The Manage Images window opens. Notice the CentOS 5 EMI in the list of images.

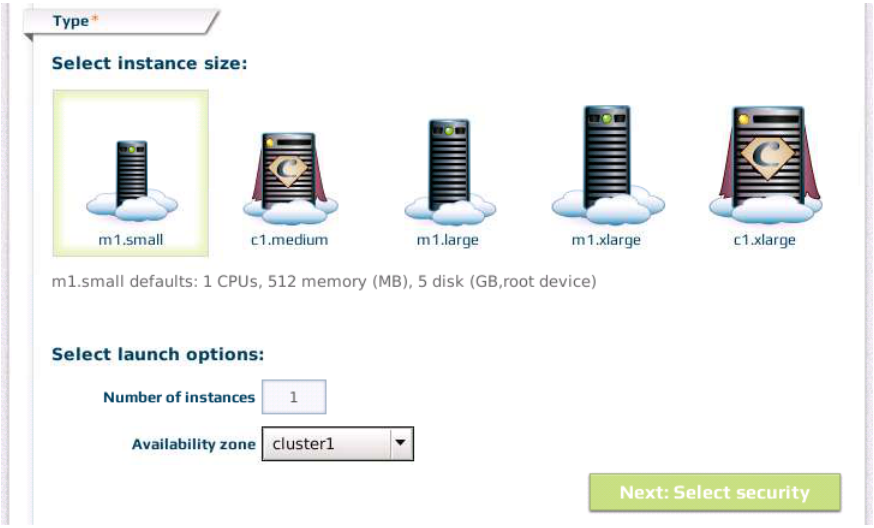


2. Click the plus symbol next to the images to find the EMI image in the *kvm-centos* bucket.

3. Click the **Launch Instance** button next to your CentOS 5 image in the *kvm-centos* bucket.

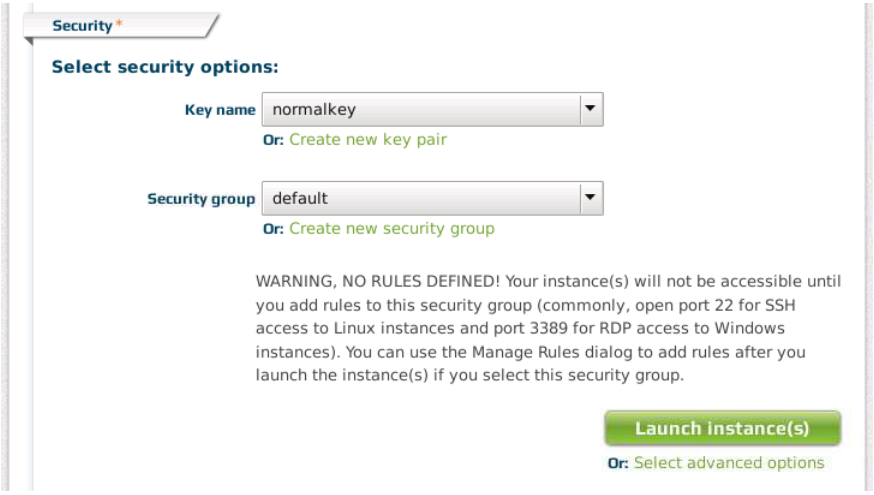
The Create New Instance window opens.

4. In the Create New Instance window, select **m1.small**. Then choose to launch **1** instance in the **cluster1** availability zone.



5. Click the **Next: Select security** button.

6. In the Create New Instances window in the **Security** section, select the **normalkey** and the security group named **default**. .

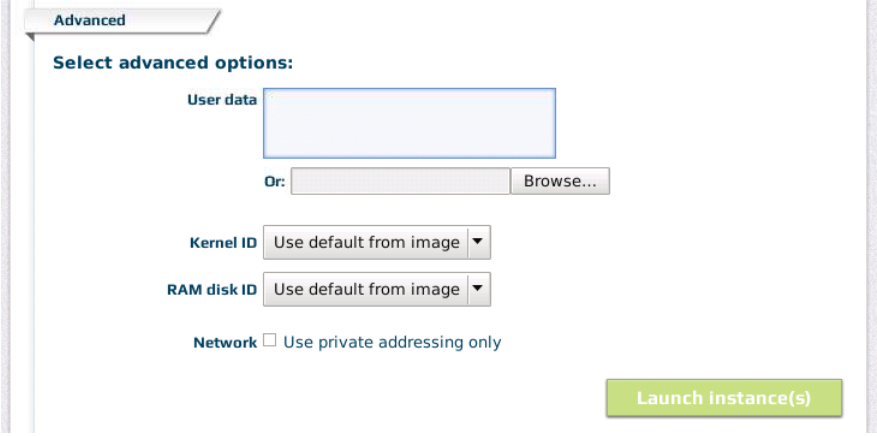


Note: Notice the warning about the default security group for the *sales* account. You will take care of this in a later lab step.

7. Click the **Select advanced options** link. Do NOT launch the instance.

8. In the Create New Images window, in the **Advanced** section, look at the available controls. User data was described and used in a another lesson. You may also choose a specific EKI or ERI to use when launching this

instance. You may also specify that this instance not receive a public IP address when it is launched. Do NOT launch the instance yet.

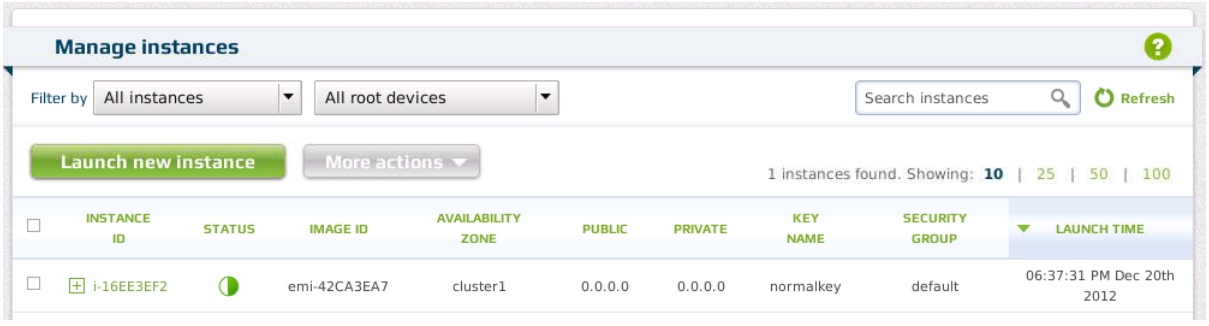


9. In the Create New Instance window, notice that the previous section labels have transformed to active screen controls. If you had the need to, you could return to any previous section and change your selections.

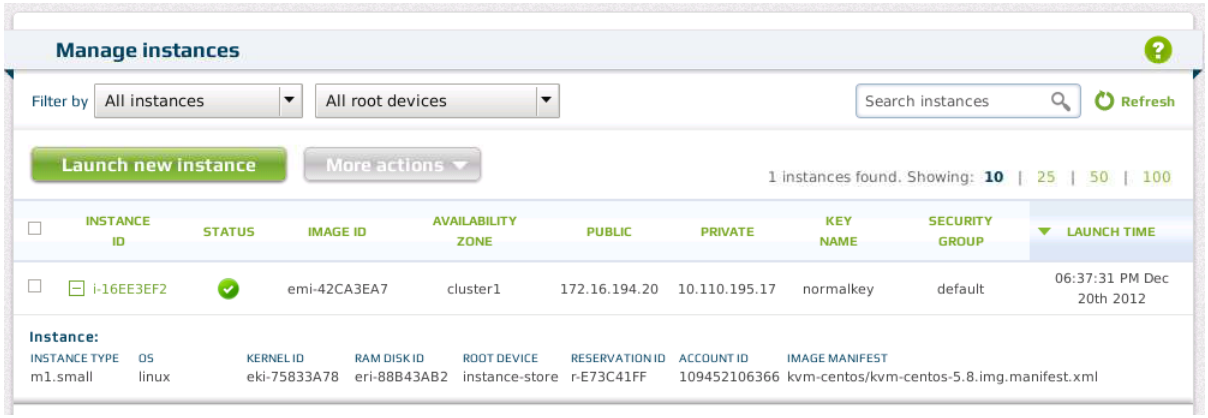
10. Notice the Summary window to the right-side of the Create New Instance window. It provides a summary of your selections as well as a **Cancel this instance** link. Do NOT cancel the instance.

11. In the Create New Instance window, click the **Launch instance(s)** button to launch the instance.

The Manage instances window opens and displays the newly launched instance. Notice the status icon indicates that the instance is pending but not yet running. The icon will change once the instance has fully booted. You can also always click the **Refresh** link in the upper right-hand corner to refresh the display.



12. Click the plus symbol next to the instance ID number. Notice the additional information about the instance displayed in the window.



Note: At the last update for this part of the book, this step did not work due to `python-boto` in `EPEL` being updated and breaking compatibility with the user console. If you get to this point and the step is not working, run the following command on the front-end machine to fix this problem:

```
yum downgrade python-boto
```

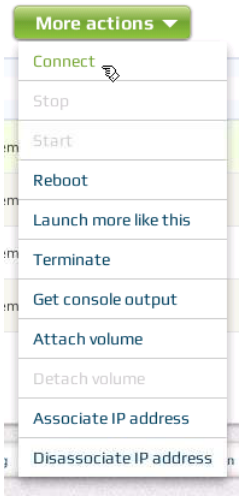
13. Click the minus symbol next to the instance ID in order to reduce the amount of displayed information for the instance.
14. Leave the instance running.

Connect to a Running Instance Using the User Console

In this section of the lab you will use the Eucalyptus User Console to log in to a running instance.

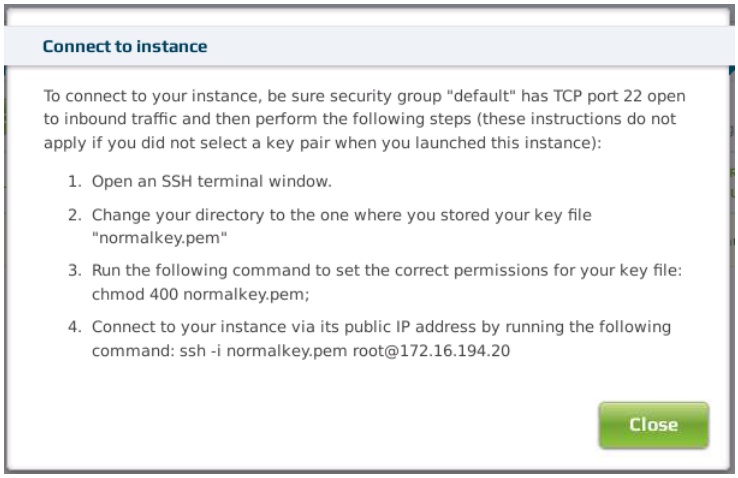
1. Desktop In the User Console in the Manage instances window, click the check box next to your running instance.

2. Click the **More Actions** button and then select **Connect** from the menu.



The Connect to instance window opens with instructions about how to connect to a running instance.

3. Read *and follow* the instructions in the Connect to instance window. Were you able to connect to your instance? Why or why not? You will fix the problem in a later lab exercise.



4. Click **Close** in the Connect to instance window.

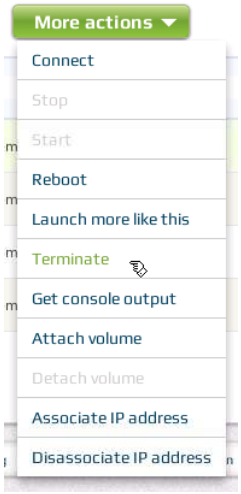
Terminate an Instance Using the User Console

Desktop

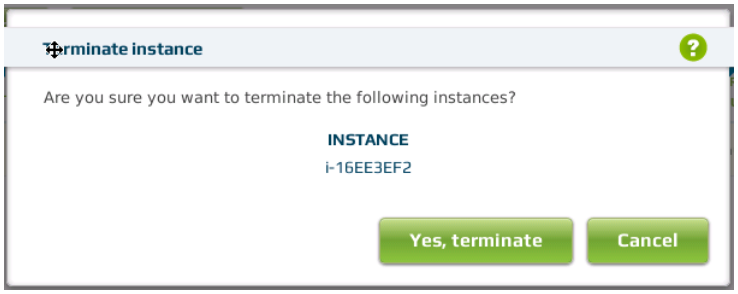
In this section of the lab you will use the Eucalyptus User Console to terminate the running instance.

1. In the User Console in the Manage instances window, ensure that the check box next to your running instance is still selected.

2. In the Manage instances window, click the **More actions** button and then select **Terminate** from the menu.



3. Confrm the termination by selecting the **Yes, terminate** button in the Terminate instance window.



4. Once the instance has been terminated, remain logged in to the User Console.

Create a security group using the User Console

Desktop

In this section of the lab exercise you will create a security group using the Eucaluptus User Console. Then you will add a firewall rule to your security group.

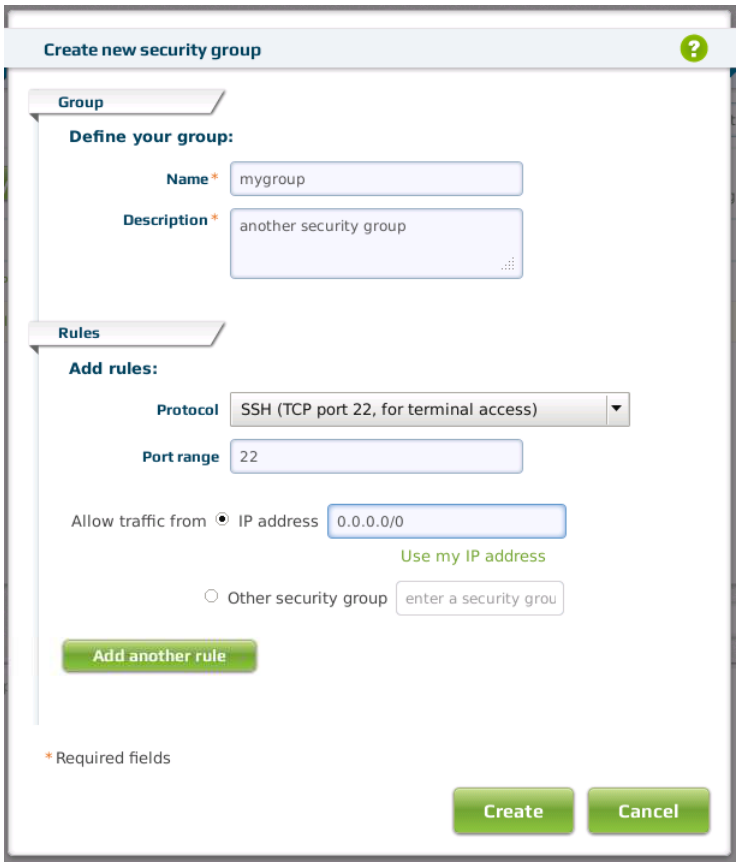
1. In the User Console, click the **Network & Security** icon and then select **Security Groups**.

The Manage security groups window opens.

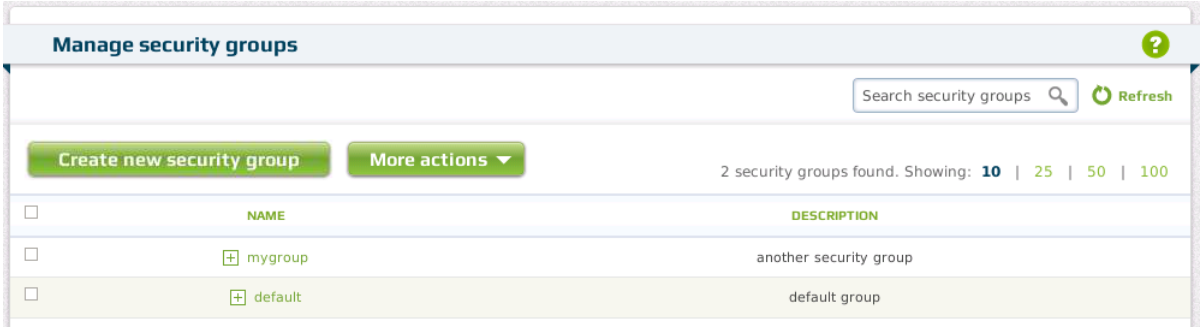
2. Click the **Create new security group** button.

The Create new security group window opens.

3. Fill out the Create new security group form. Name the security group `mygroup` and provide the description of `another security group`. Select **SSH (TCP port 22, for terminal access)** for the Protocol. Allow access from the network `0.0.0.0/0`. Click **Create**.

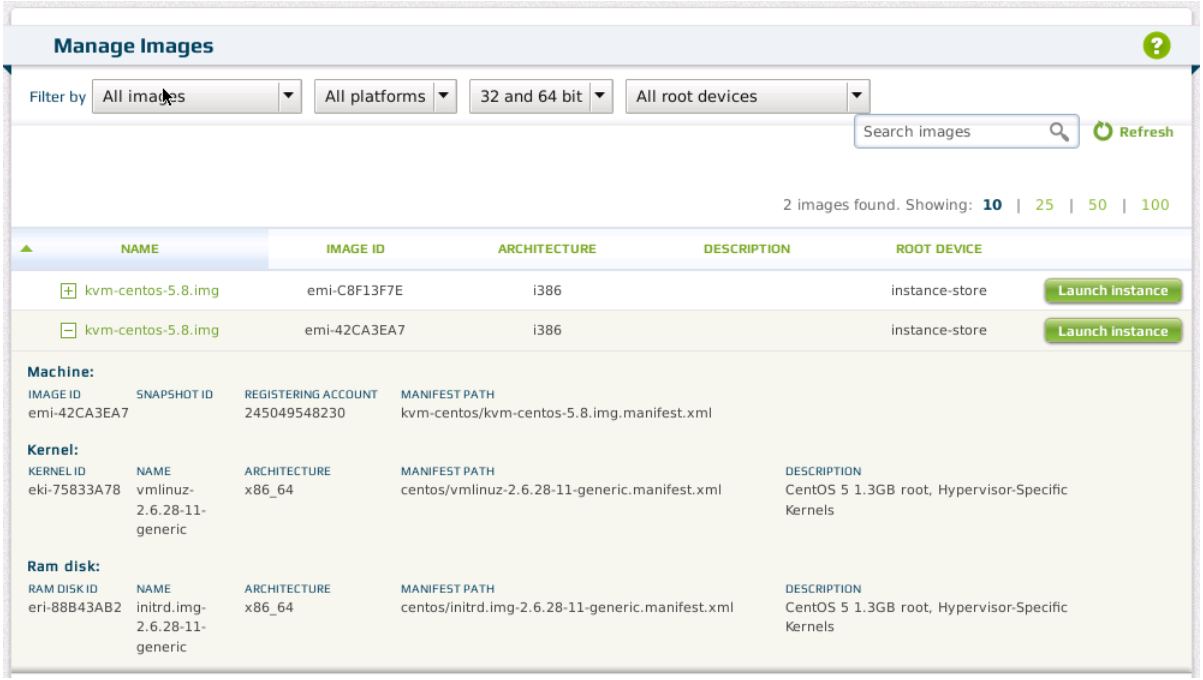


The new security group is listed in the Manage security groups window.

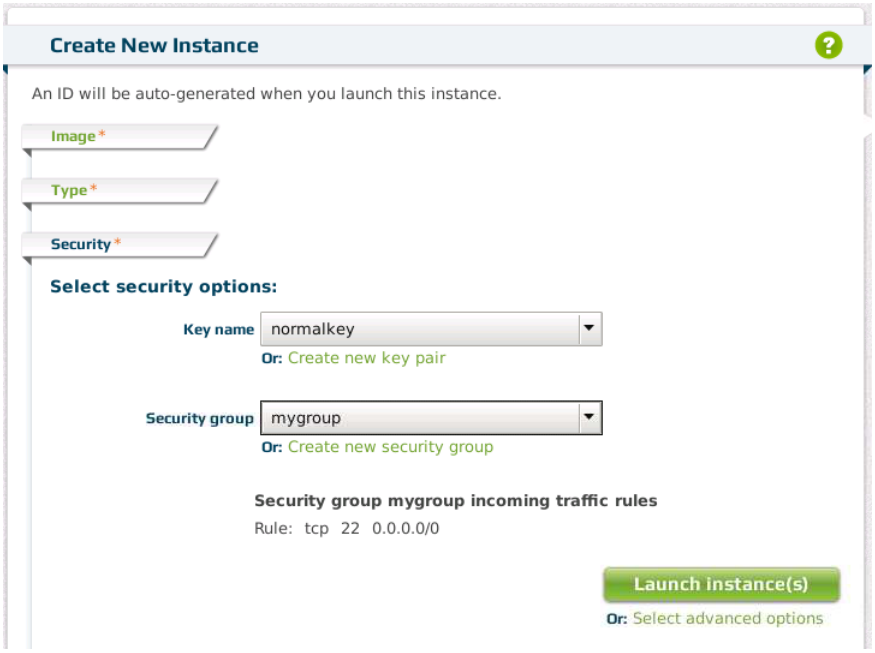


4. Click the **Images** icon to navigate to the Manage images window.

5. Click the **Launch instances** button next to your EMI image that is in the *kvm-centos* bucket.



6. Accept the default vmtype, number of instances, and availability zone and click the **Next: Select security** button.
7. Accept the default key pair and select your **mygroup** security group from the drop-down menu. Then click the **Launch instance(s)** to launch the instance. Allow the instance to reach a running status.



8. Remain logged in to the User Console.
9. Open an xterm window on the Debian desktop and then use SSH to log in to your running instance.

```
#ssh -i <key_file> <instance_public_IP>
```

Did it work? You might have received a security warning message from the `ssh` command. If you did, the problem is that the instance's IP address was formerly used by a previous instance and its security key information is cached in the `/root/.ssh/known_hosts` file. You will need to edit this file and remove the entry associated with your current instance's IP address. You will have to do this each time the IP address is reused by a new instance. While it is less secure, you could also create an shell alias for the `ssh` command in the shell startup file, for example the `.bashrc` file. The alias would have the syntax `alias ssh='ssh -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no'`.

10. Exit from the instance and close the xterm window.

```
# exit
```

Modify a security group using the User Console

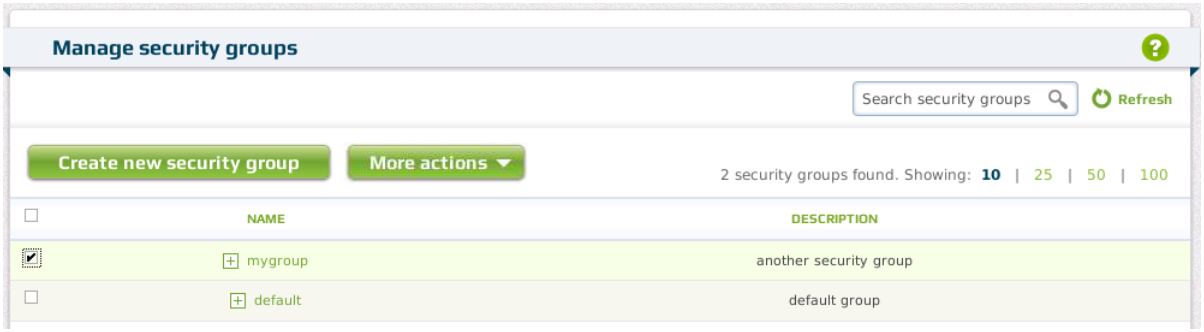
Desktop

In this section of the lab exercise you will use the Eucalyptus User Console to modify the firewall rules associated with a security group.

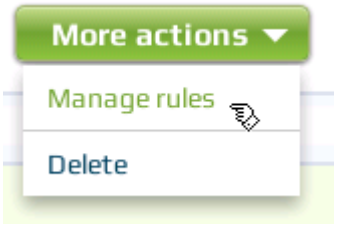
1. On the Debian desktop in the User Console window, click the **Network & Security** icon and then select **Security Groups**.

The Manage security groups window opens.

2. Click the check box next to your *mygroup* security group in order to select it.

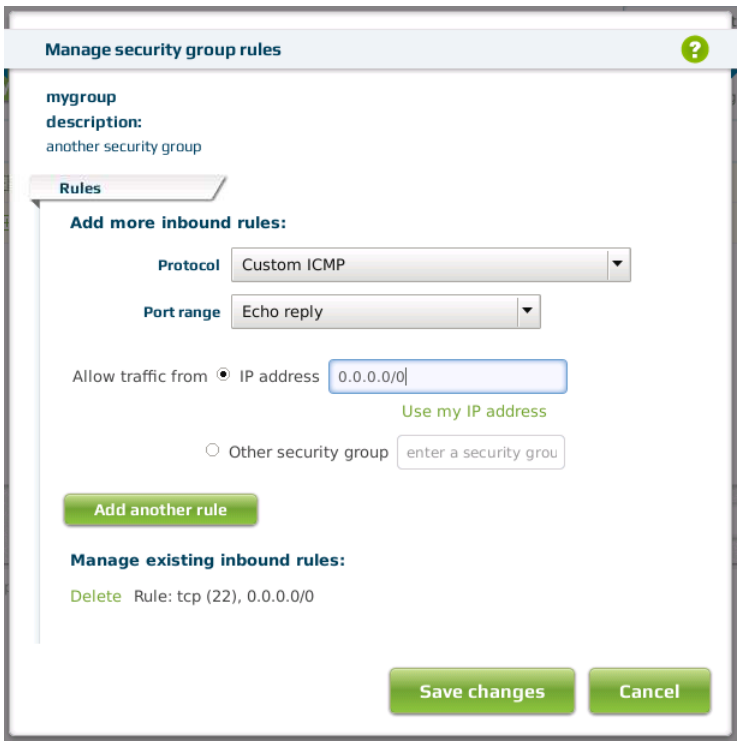


3. Click **More actions** and then select **Manage rules** from the menu.

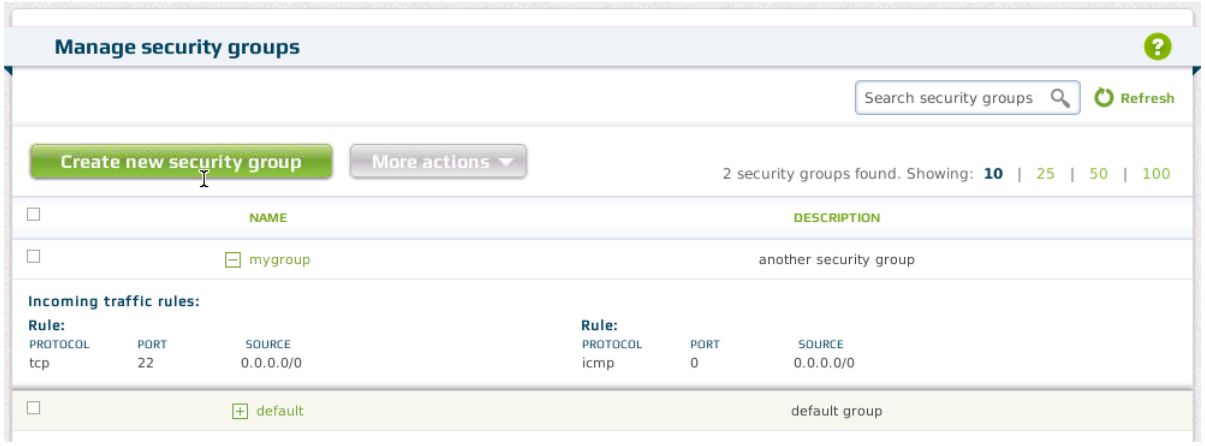


The Manage security group rules window opens.

4. In the Manage security group rules window, select **Custom ICMP** as the protocol, **Echo reply** for the port range, and allow traffic from `0.0.0.0/0`. Then click the **Save changes** button.



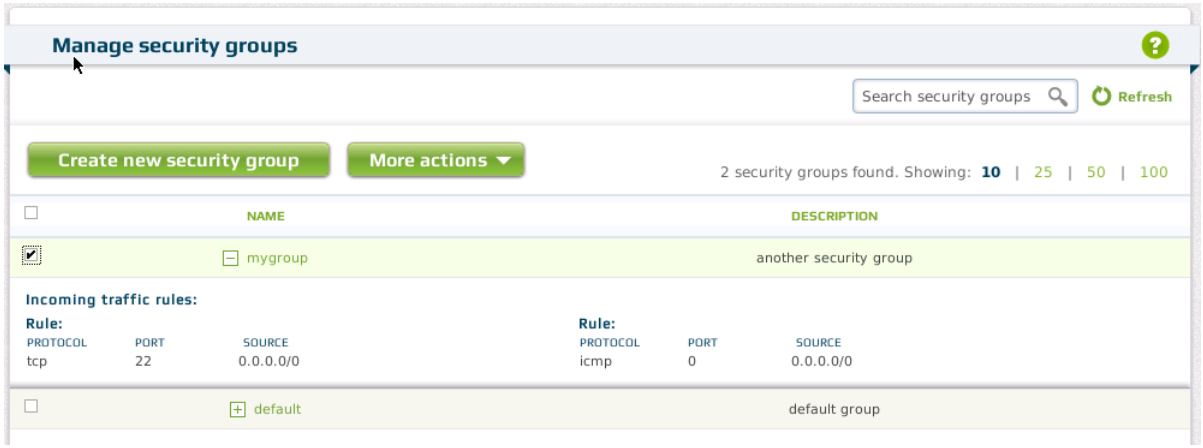
5. Click the plus symbol next to your security group in order to see its firewall rules. You might have to click **Refresh** in order to see the newest rule.



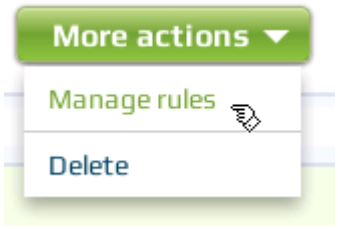
6. From the Debian desktop, open an xterm window and `ping` the public IP address of your running instance.

```
# ping <instance_public_IP>
```

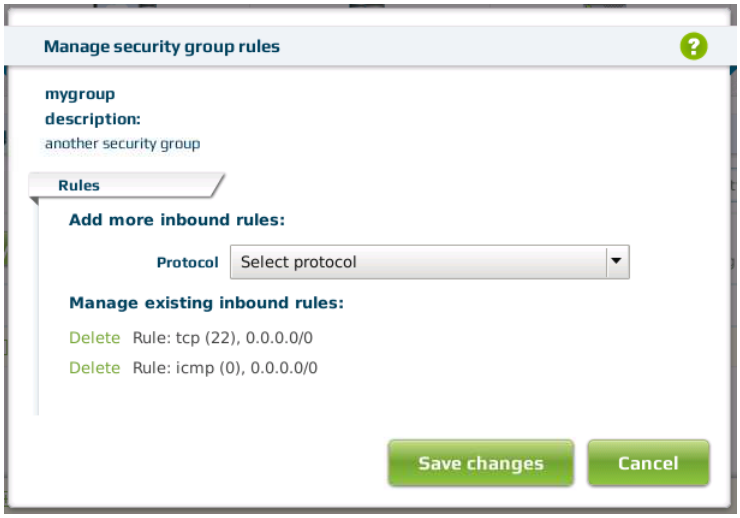
   Did it work? It should have. Use Ctrl-C to stop the `ping` command.
7. Close the xterm window on the Debian desktop (where you ran the `ping` command).
8. In the User Console, select the check box again next to your security group.



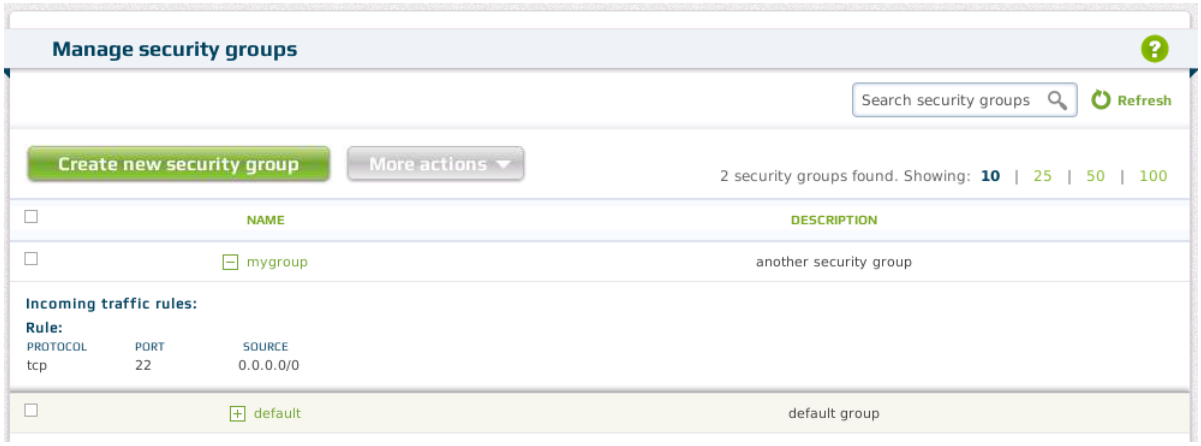
9. Select **Manage rules** from the **More actions** button menu.



10. Click **Delete** in front of the ICMP rule. Then click the **Save changes** button.



11. Click the plus symbol next to your security group in order to see its rules. You might have to click **Refresh** again to view the latest changes.



Has the ICMP rule been removed?

12. Remain logged in to the User Console.

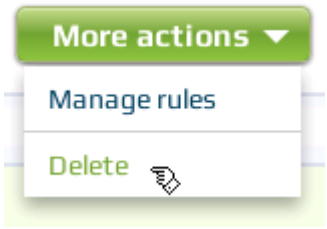
Delete a security group using the User Console

Desktop

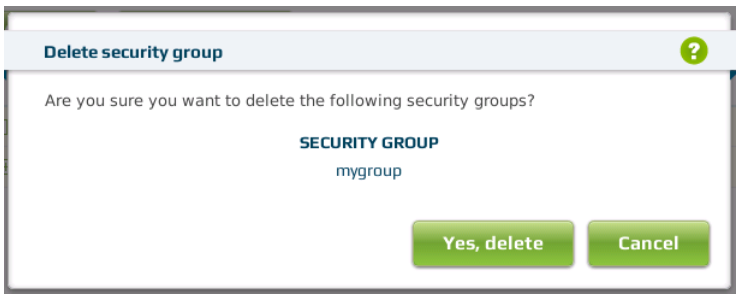
In this section of the lab exercise you will delete a security group using the Eucalyptus User Console.

1. In the User Console, select the check box next to your security group.

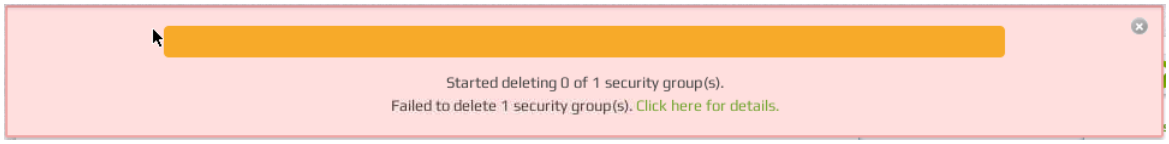
2. Click **More actions** and select **Delete** from the menu.



3. Confirm the deletion by clicking the **Yes, delete** button.



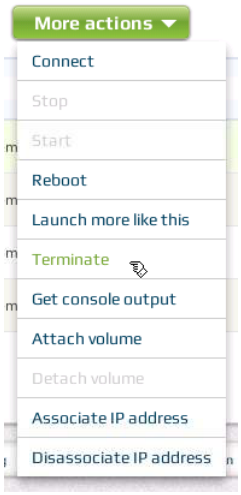
4. The deletion should have failed and you should have recieved an error message. Click **Click here for details.** The security group could not be deleted because it contains a running instance.



5. Click the **X** symbol to dismiss the error message window.
6. Click the **Instances** icon to open the Manage instances window.

7. Click the check box next to your instance in order to select it.

8. Select **Terminate** from the **More actions** menu. Then click **Yes, terminate** to confirm the termination in the Terminate instance window.

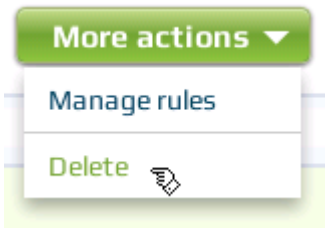


9. Click the **Network & Security** icon and then select **Security Groups**.

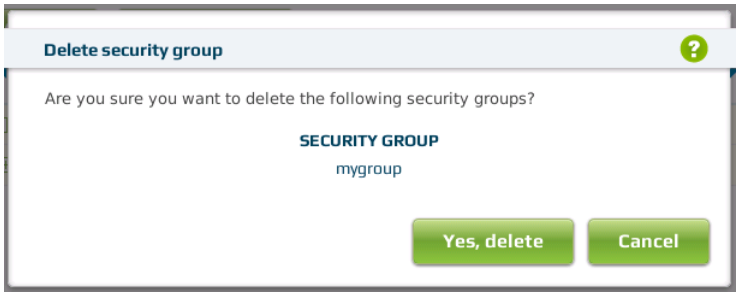
The Manage security groups window opens.

10. Select the check box next to your security group.

11. Click **More actions** and select **Delete** from the menu.



12. Confirm the deletion by clicking the **Yes, delete** button.



Did it work this time? It should have.

Reserve an elastic IP address using the User Console

Desktop

In this section of the lab you will use the Eucalyptus User Console to reserve a public IP address from the pool of public IP addresses available in your cloud.

1. In the User Console click the **Network & Security** icon and then select **IP Addresses** from the drop-down menu.

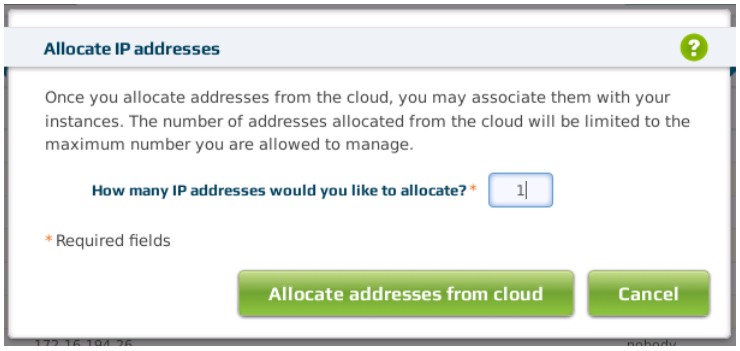
The Manage IP addresses window appears.

2. View the Manage IP addresses window information. Because you have not reserved any IP addresses at this point, you should not see any IP addresses listed.

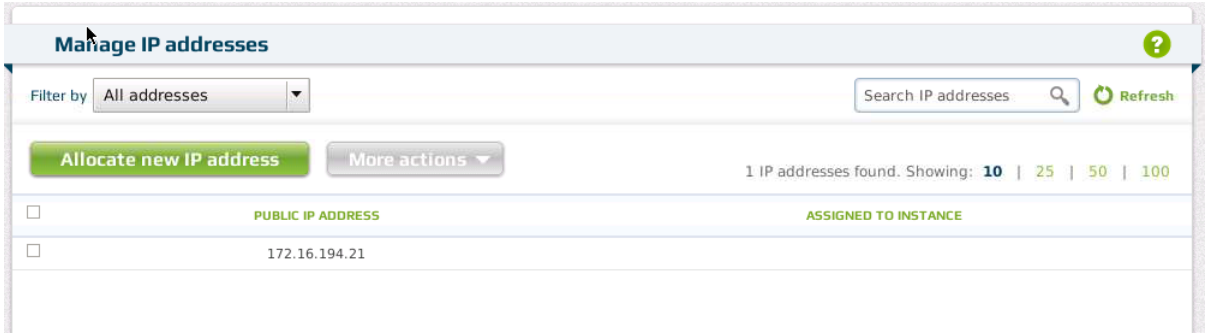
3. In the Manage IP addresses window, click the **Allocate new IP address** button.

The Allocate IP addresses window opens.

4. Type `1` in the **How many IP addresses would you like to allocate** text box. The click the **Allocate addresses from cloud** button.



5. View the results in the Manage IP addresses window. One IP address should be reserved for user *admin*.



6. Remain logged in to the User Console.

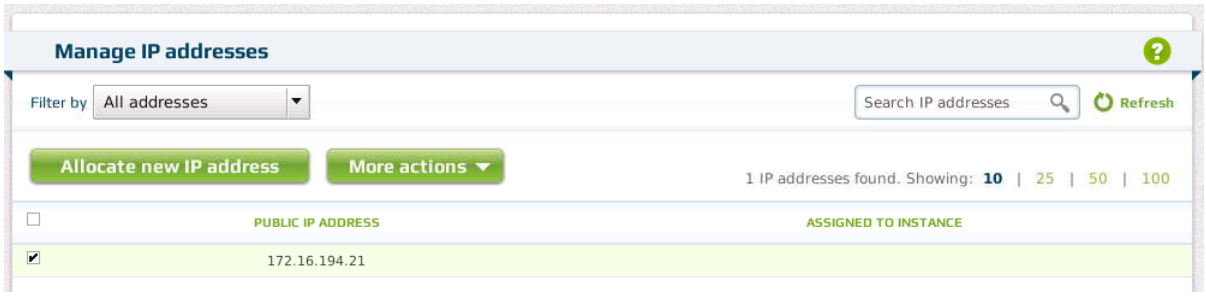
Assign an elastic IP address using the User Console

Desktop

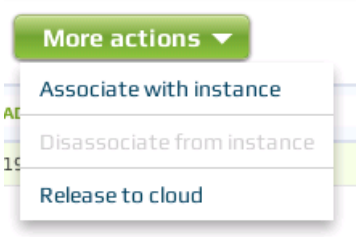
In this section of the lab you will use the Eucalyptus User Console to assign a reserved public IP address to a specific instance.

1. In the User Console, click the Instances icon. Write down the instance ID number of your running instance. You will need this ID number in a later lab step.

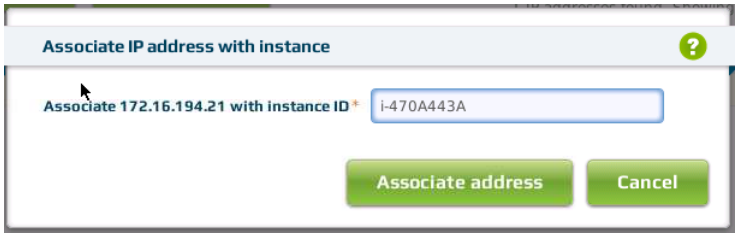
2. In the User Console, click the check box next to your reserved IP address.



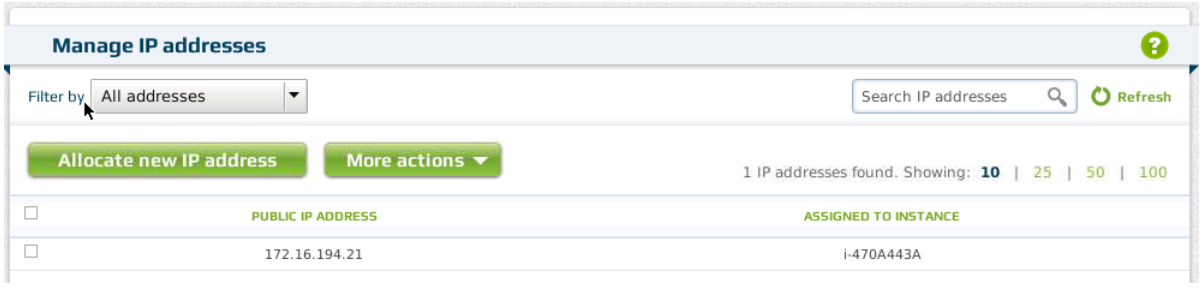
3. Select **Associate with instance** on the **More actions** menu.



4. In the window that opens, begin to enter the instance ID number that you wrote down earlier. A list of instance IDs should appear. Select your instance ID from the list and then click **Associate address**.



5. View the Manage IP addresses window. You might have to click **Refresh** a few times to see the change. Which IP address is now associated with the running instance? Is this your reserved IP address?



6. On the Debian desktop, open an xterm window and SSH into the running instance.

```
# ssh -i <key_file> <instance_public_IP>
```

Note: If this step fails, check to make sure port 22 has been enabled for the *default* security group. This should have been done in an earlier lab.

7. Run the `ifconfig` command inside the running instance. What IP address is reported? Is this the private or public IP address?

```
# ifconfig
```

8. Exit from the instance and close the xterm window.

```
# exit
```

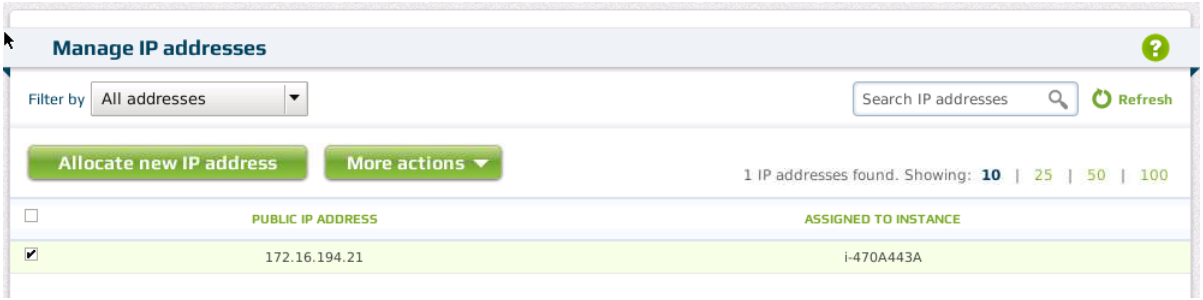
9. Remain logged in to the User Console.

Unassign an elastic IP address using the User Console

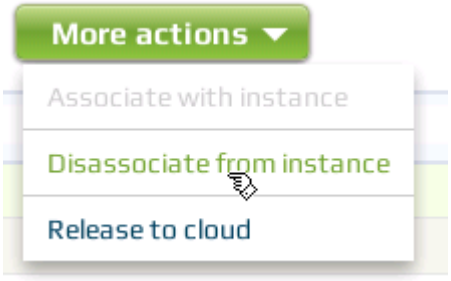
Desktop

In this section of the lab you will use the User Console to disassociate a reserved public IP address from a specific instance.

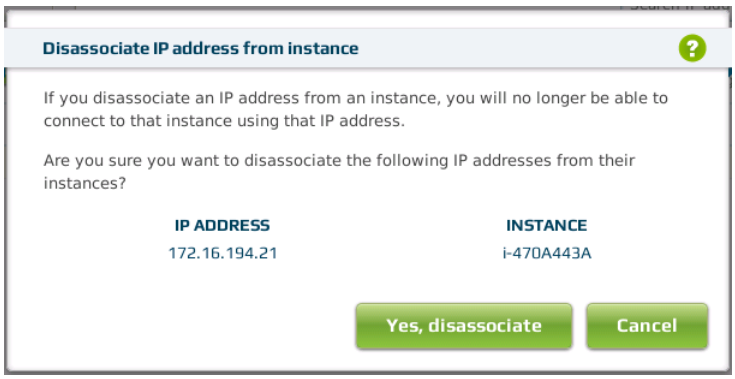
1. In the User Console, in the Manage IP addresses window, select the check box next to the reserved and assigned IP address.



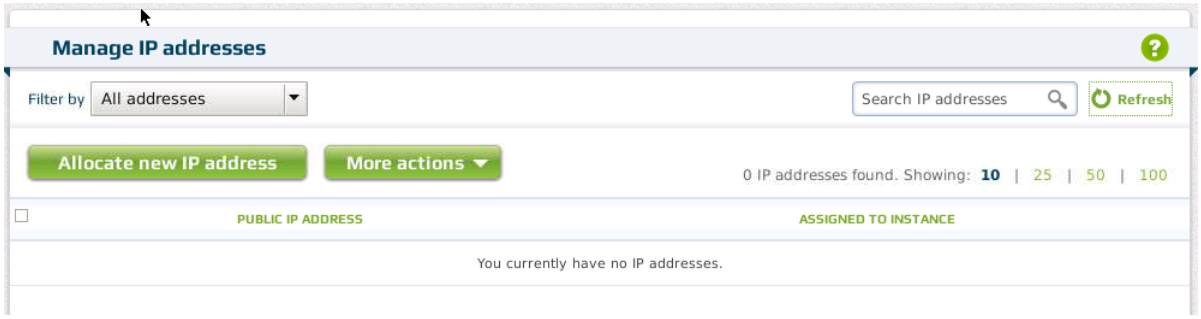
2. Select **Disassociate from instance** from the **More actions** menu.



3. Select **Yes, disassociate** in the Disassociate IP address from instance window that appears.



4. View the Manage IP addresses window. What happened to the reserved IP address?



5. On the Debian desktop, open an xterm window and SSH into the running instance. Which IP address did you have to use in order to be successful?

```
# ssh -i <key_file> <instance_public_IP>
```

6. Run the `ifconfig` command inside the running instance. What IP address is reported?

```
# ifconfig
```

7. Exit from the instance and close the xterm window.

```
# exit
```

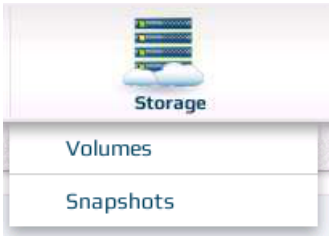
8. Remain logged in to the User Console.

Create a storage volume using the User Console

Desktop

In this section of the lab you will create a storage volume using the Eucalyptus User Console.

1. In the User Console click **Storage** and select **Volumes** from the menu.

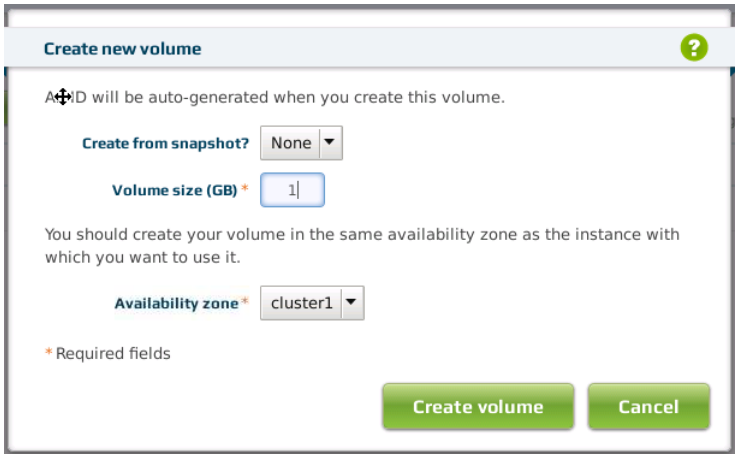


The Manage volumes window opens.

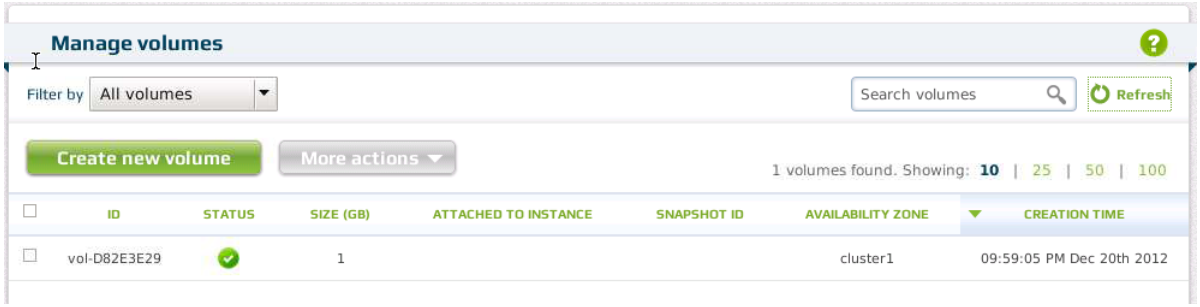
2. In the Manage volumes window, click the **Create new volume** button.

The Create new volume window opens.

3. In the Create new volume window, create a 1GB volume in *cluster1* that is not from a snapshot. Then click **Create volume**.



4. Note the Manage volumes window displays the created volume.



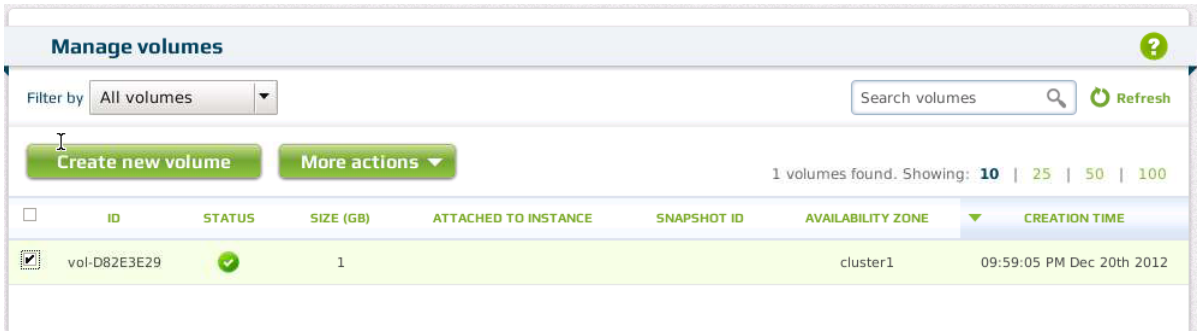
5. Remain logged in to the User Console.

Attach a volume to an instance using the User Console

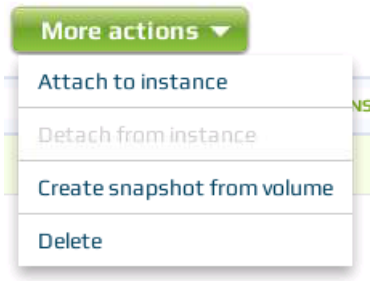
Front End

In this section of the lab you will attach a volume to a running instance using the Eucalyptus User Console.

1. In the User Console in the Manage volumes window, click the check box next to the volume to select the volume.

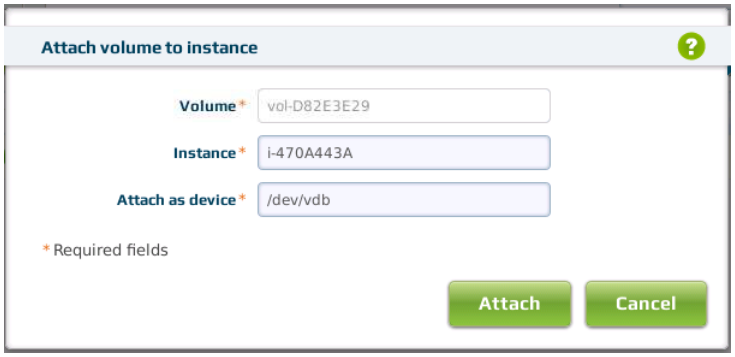


2. In the Manage volumes window, click **More actions** and select **Attach to instance**.



The Attach volume to instance window opens.

3. In the Attach volume to instance window, type `i-` to open a drop-down menu with a list of running instances. Select your running instance from the list. Change the default device name to `/dev/vdb`, and click **Attach**.

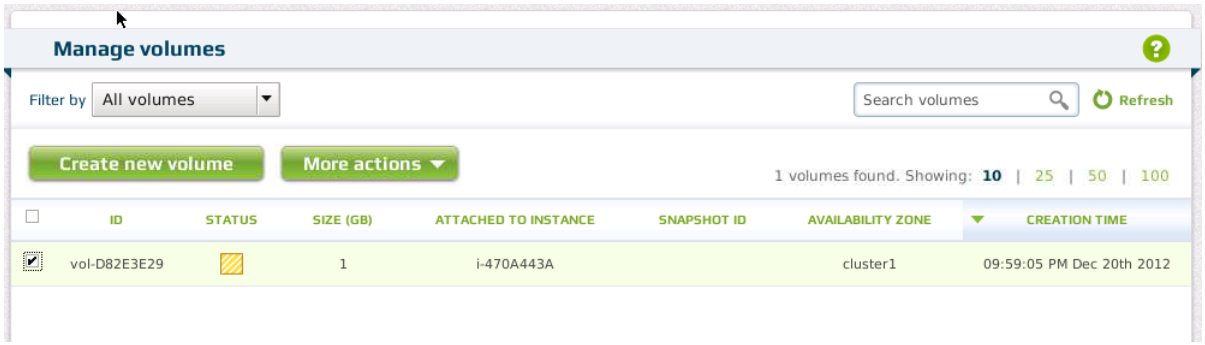


4. In the Manage volumes window, notice that the volume is attached to the instance. You might need to click **Refresh** to see the updated information.
5. Remain logged in to the User Console.

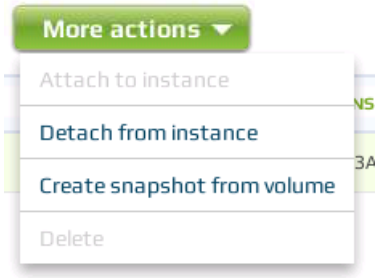
Detach a volume using the User Console

In this section of the lab you will detach the volume from your instance using the Eucalyptus User Console.

1. In the User Console in the Manage volumes window, select the check box next to your attached volume.

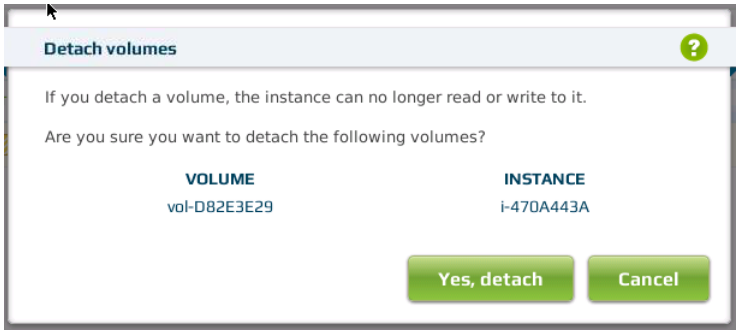


2. In the Manage volumes window, click **More actions** and select **Detach from instance**.

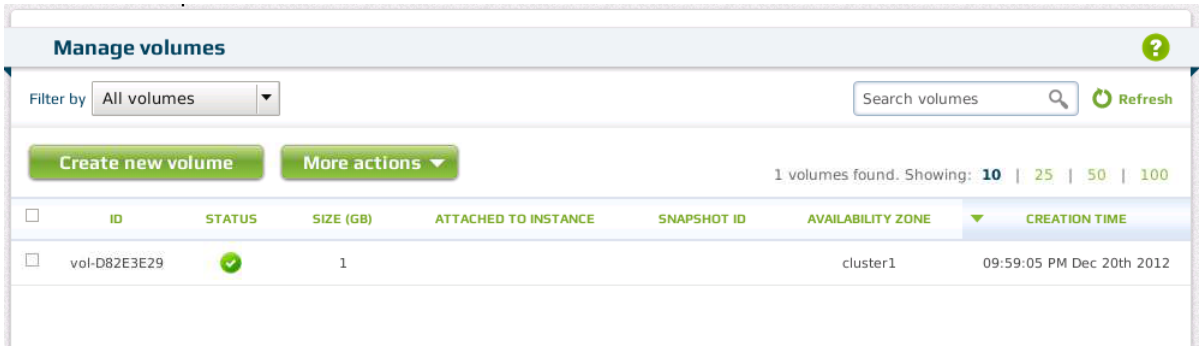


The Detach volume window opens.

3. In the Detach volume window, click **Yes, detach** to confirm the operation.



4. In the Manage volumes window, notice that the volume is no longer attached to an instance.

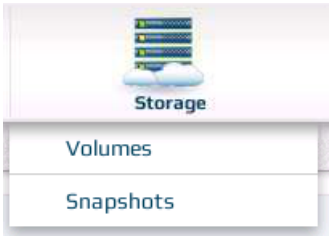


5. Remain logged in to the User Console.

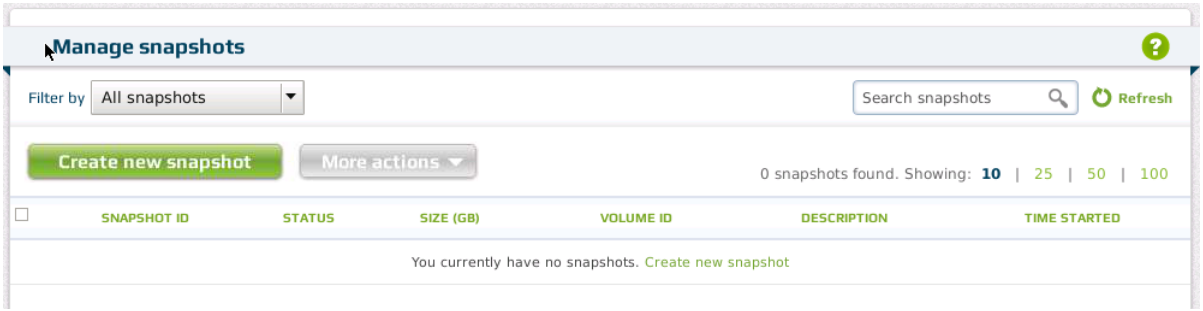
Snapshot a volume using the User Console

In this section of the lab you will create a snapshot of your volume using the Eucalyptus User Console.

1. In the User Console click **Storage** and then select **Snapshots**.



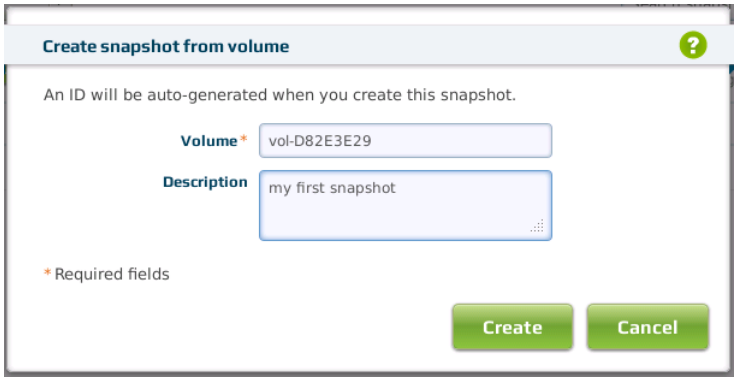
The Manage snapshots window opens.



2. In the Manage snapshot window, click the **Create new snapshot** button.

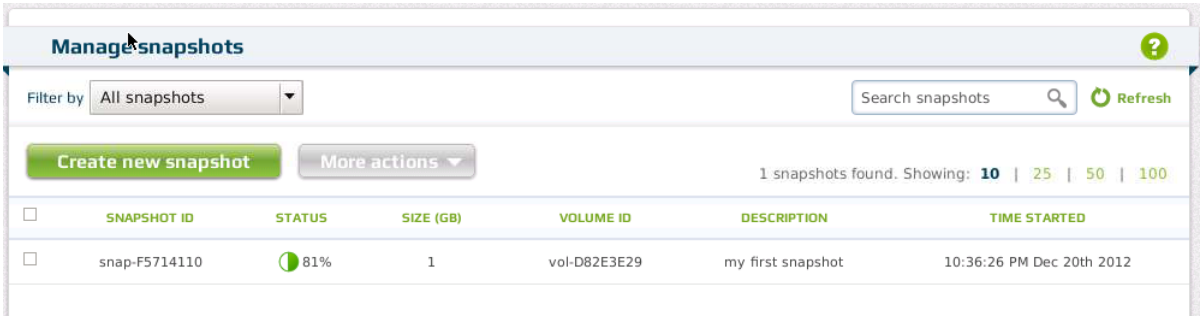
The Create snapshot from volume window opens.

3. In the Create snapshot from volume window, type `vol-` to open the drop-down menu with a list of volumes. Select your volume, type a description of `my first snapshot`, and click **Create**.



Snapshot creation begins.

4. In the Manage snapshots window, monitor snapshot creation progress until it reaches 100%.



5. Remain logged in to the User Console.

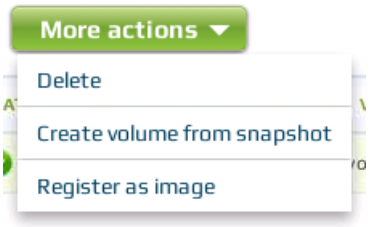
Create a volume from a snapshot using the User Console

Front End

In this section of the lab you will use the Eucalyptus User Console to create a new volume using the snapshot taken in a previous section of this lab.

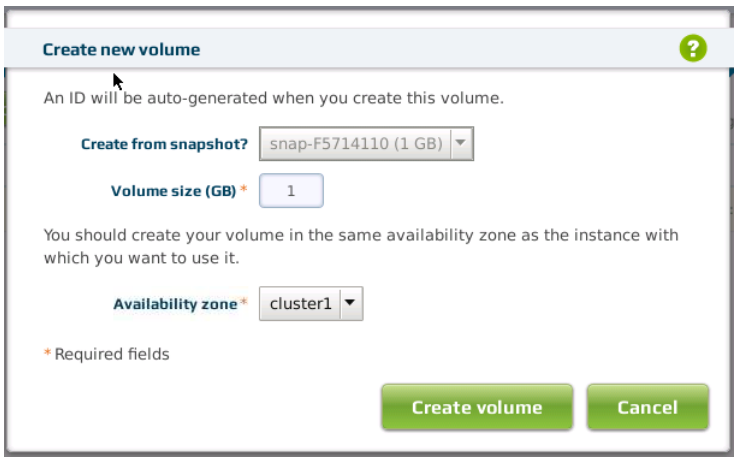
1. In the User Console in the Manage snapshot window, select the check box next to your snapshot in order to select it.

2. In the Manage snapshots window, click **More actions** and select **Create volume from snapshot**.

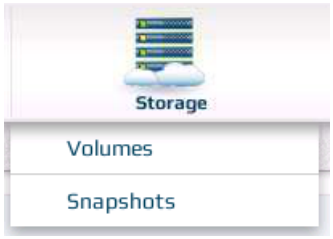


The Create new volume window opens.

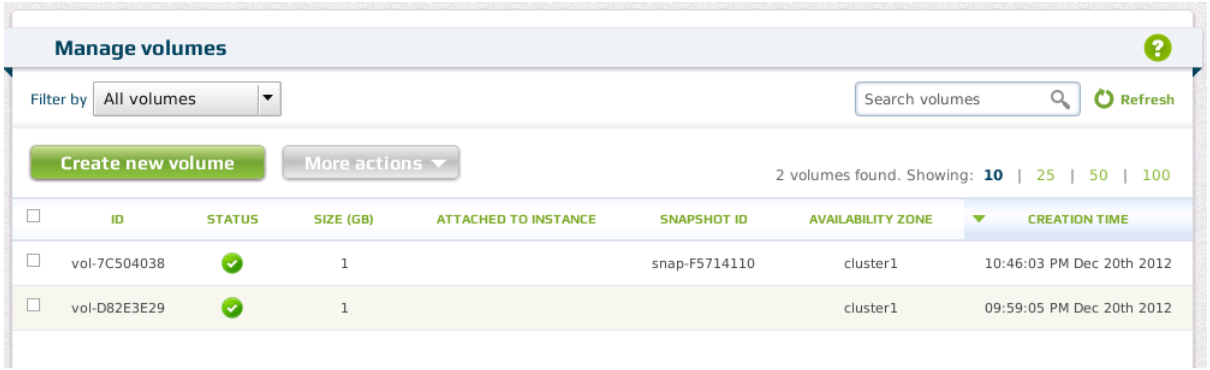
3. In the Create new volume window, accept the default choices and click **Create volume**.



4. In the User Console, click the **Storage** icon and select **Volumes** to see the new volume.



The new volume is displayed.



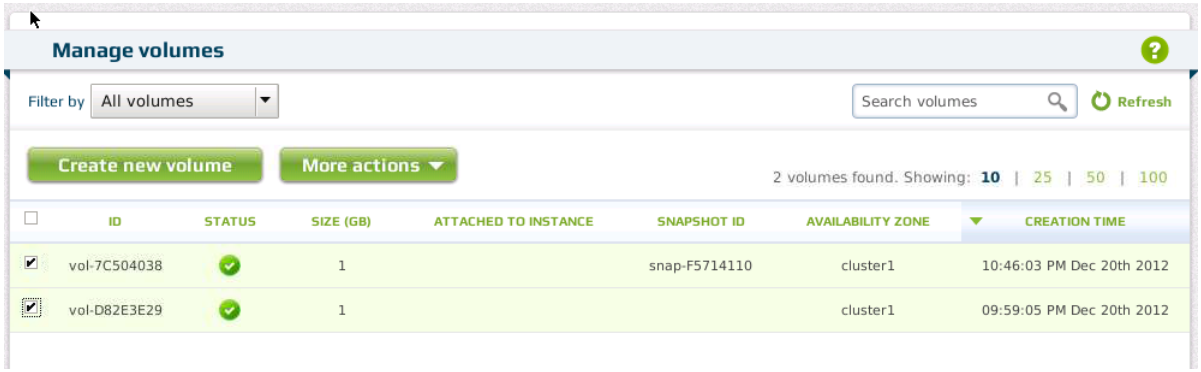
5. Remain logged in to the User Console.

Delete a volume and snapshot using the User Console

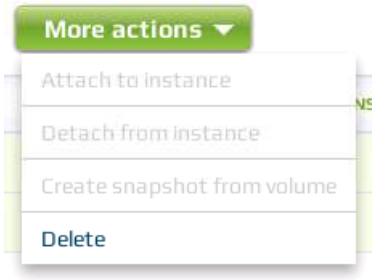
Front End

In this section of the lab you will delete volumes and a snapshot using the Eucalyptus User Console. You will also terminate the running instance.

1. In the User Console in the Manage volumes window, click the check boxes next to the two volumes.

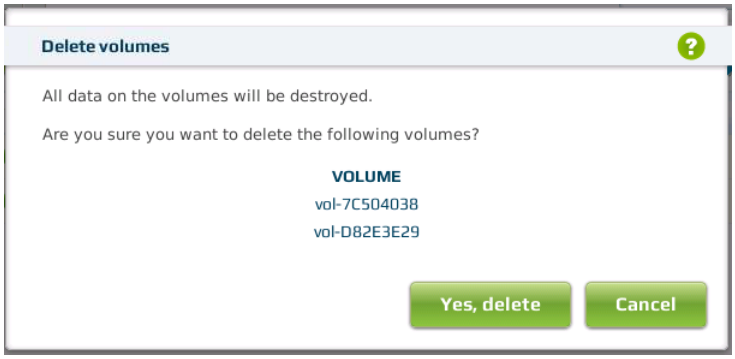


2. In the Manage volumes window, click **More actions** and select **Delete** to delete the volumes.

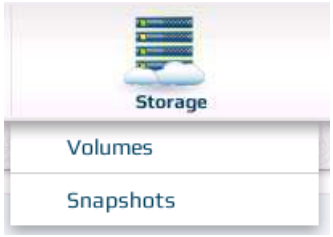


The Delete volumes window opens.

3. In the Delete volumes window, click **Yes, delete** to confirm the delete operation.



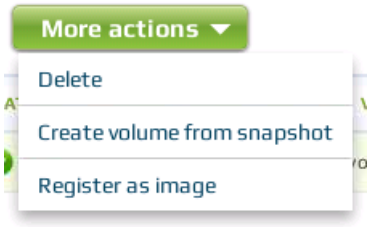
4. In the User Console, click **Storage** and select **Snapshots**.



The Manage snapshots window opens.

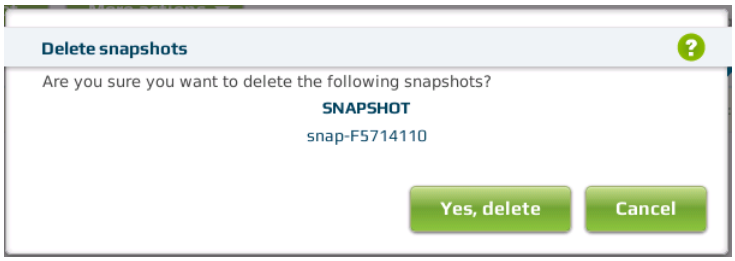
5. In the Manage snapshots window, click the check box next to the snapshot in order to select it.

6. In the Manage snapshots window, click **More actions** and select **Delete**.



The Delete snapshots window opens.

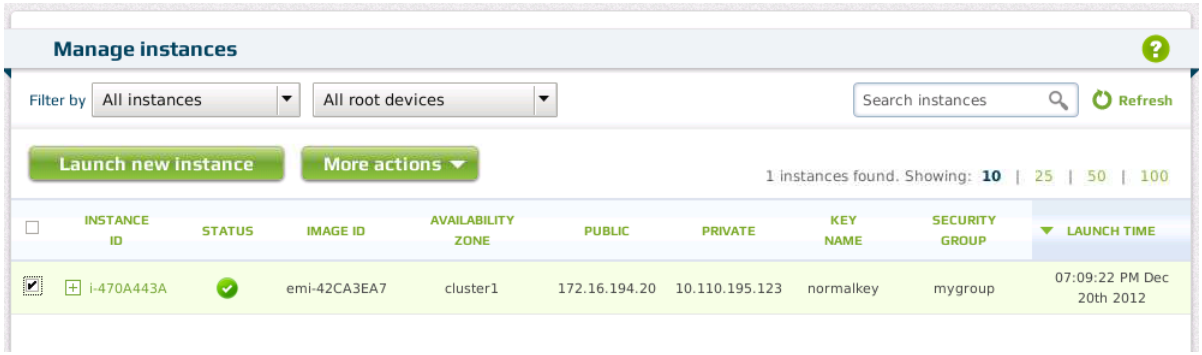
7. In the Delete snapshots window, select **Yes, delete** to confirm the delete operation.



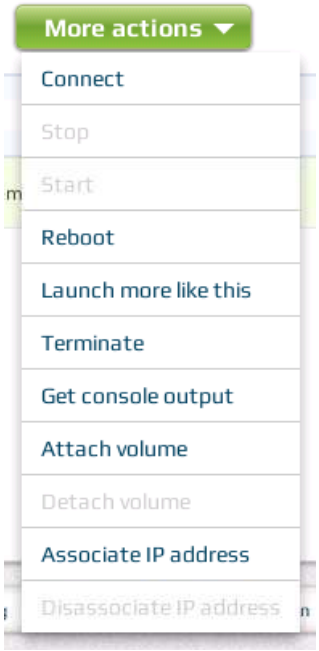
8. In the User Console, click the **Instances** icon.

The list of running instances is displayed in the Manage instances window.

9. In the Manage instances window, click the check box next to your running instance.

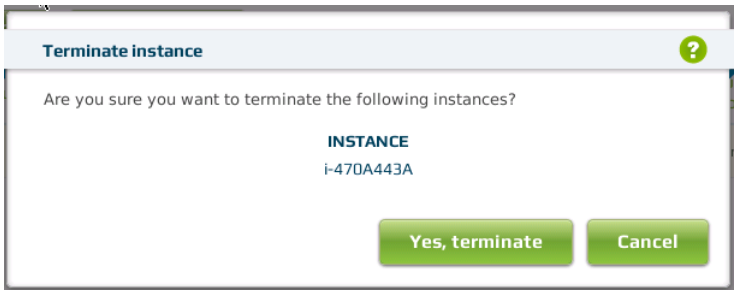


10. In the Manage instances window, click **More actions** and select **Terminate** to terminate the instance.

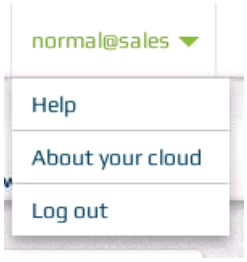


The Terminate instance window opens.

11. In the Terminate instance window, click **Yes, terminate** to terminate the running instance.



12. Log out of the User Console.


normal@sales
Help
About your cloud
Log out

Monitor Eucalyptus Overview

The Eucalyptus Dashboard provides a usage reporting function that helps you monitor and report on your cloud deployment. This section will show you how to generate, view, interpret, and save usage reports in the Eucalyptus Dashboard. It will also touch on the wide array of third-party monitoring tools that are available to extend this capability.

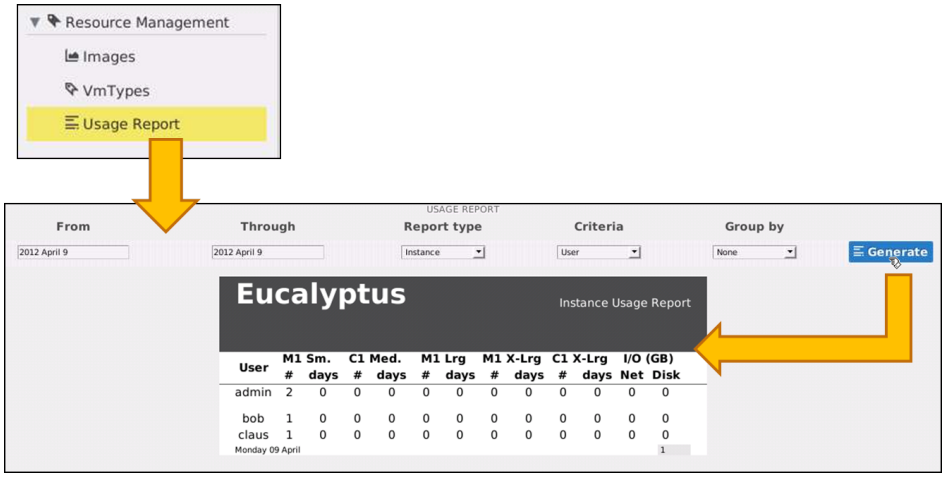
Dashboard Usage Reports

The Eucalyptus Dashboard reports cloud resource usage over a user-specified date range. Three types of usage reports are available:

- Instance reports, which report the number of instances, the length of time each one has been running, and network and storage I/O usage for each
- Storage reports, which report volume and snapshot storage space usage
- S3 (Walrus) reports, which report the maximum number of buckets as well as storage space usage

Generate Usage Reports

To generate a usage report, click the **Usage Report** button in the QUICK LINKS tab. Then select the date range, type of report, sort criteria, and how you want to group the report output. Finally, click **Generate**.



When you select a range of dates for the report, the report is inclusive of the starting and ending dates.

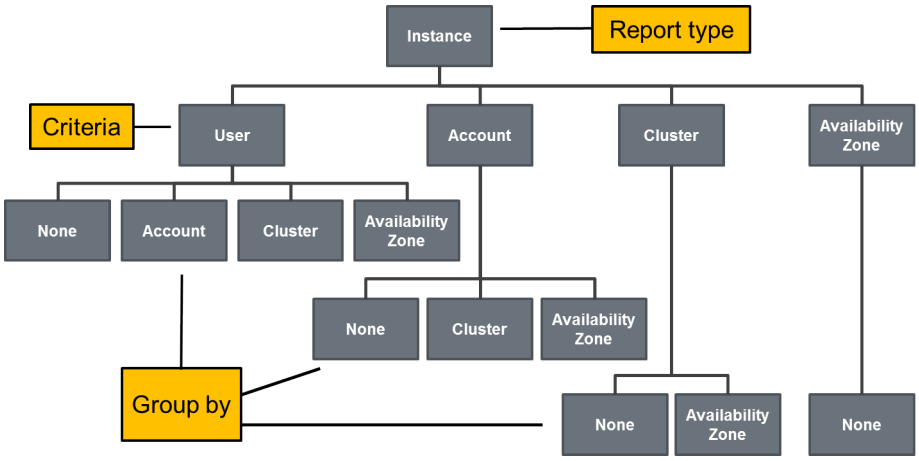
Report type is a drop-down menu that include **Instance**, **Storage**, and **S3** choices.

Criteria is a drop-down menu whose choices depend on the type of report selected. **Criteria** is used to sort the data in the report.

Group by is drop-down menu whose choices depend on the type of report and the criteria selected. **Group by** is used to organize the sorted data in different ways. For example, a list of data sorted by users could be further organized so that all the users are listed by their account, or by the availability zone in which they operate.

Instance Report Options

Because instances run in an availability zone and are controlled by a Cluster Controller, instance usage can be reported by user, account, cluster (Cluster Controller), and availability zone.



If **Instance** and **Account** are selected, then the **Group by** menu only has selections for **None**, **Cluster**, and **Availability Zone**.

If **Instance** and **Cluster** are selected, then the **Group by** menu only has selections for **None** and **Availability Zone**.

If **Instance** and **Availability Zone** are selected, then the **Group by** menu only has a selection option of **None**.

Instance Report Examples

This first example shows an instance report where **Criteria** was set to **User**, and **Group by** was set to **None**.

Criteria: User
Group by: None

-just a list of users

Eucalyptus Instance Usage Report

User	M1 Sm. #	M1 Sm. days	C1 Med. #	C1 Med. days	M1 Lrg #	M1 Lrg days	M1 X-Lrg #	M1 X-Lrg days	C1 X-Lrg #	C1 X-Lrg days	I/O (GB) Net	I/O (GB) Disk
admin	2	1	0	0	0	0	0	0	0	0	0	0
bob	1	0	0	0	0	0	0	0	0	0	0	0
claus	1	0	0	0	0	0	0	0	0	0	0	0

Tuesday 10 April 1

The next example shows an instance report where **Criteria** was again set to **User**, but this time **Group by** was set to **Account**.

Criteria: User
Group by: Account

-a list of users further grouped by accounts

Eucalyptus Instance Usage Report

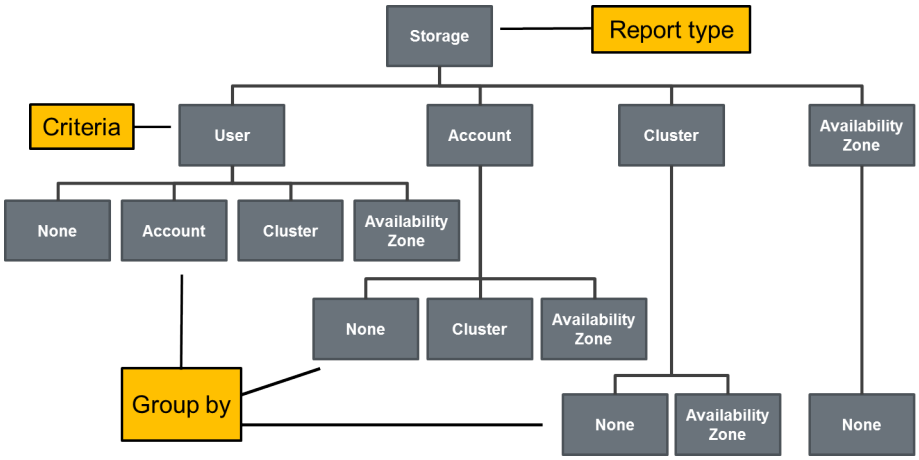
User	M1 Sm. #	M1 Sm. days	C1 Med. #	C1 Med. days	M1 Lrg #	M1 Lrg days	M1 X-Lrg #	M1 X-Lrg days	C1 X-Lrg #	C1 X-Lrg days	I/O (GB) Net	I/O (GB) Disk
eucalyptus												
admin	2	1	0	0	0	0	0	0	0	0	0	0
sales												
bob	1	0	0	0	0	0	0	0	0	0	0	0
claus	1	0	0	0	0	0	0	0	0	0	0	0

Tuesday 10 April 1

In these example reports, the user admin in the eucalyptus account has had two instances running for at least one day. Together, those instances have generated less than 1GB of either network or disk I/O. The users bob and claus in the sales account have had one instance each running for less than a day. Neither of the instances have generated 1GB of network or disk I/O up to this point in time.

Storage Report Options

Volumes and snapshots are associated with a Cluster Controller and created within an availability zone. Volume and snapshot usage can be reported by user, account, cluster (Cluster Controller), and availability zone.



If **Storage** and **Account** are selected, then the **Group by** menu has only selections for **None**, **Cluster**, and **Availability Zone**.

If **Storage** and **Cluster** are selected, then the **Group by** menu has only selections for **None** and **Availability Zone**.

If **Storage** and **Availability Zone** are selected, then the **Group by** menu only has a selection option of **None**.

Storage Report Examples

This first example shows a storage report where **Criteria** was set to **Account**, and **Group by** was set to **None**.

Criteria: Account
Group by: None

-just a list of accounts

Eucalyptus Storage Report

Account	Volumes		Snapshots	
	Max (GB)	GB-days	Max (GB)	GB-days
eucalyptus	0	0	0	0
sales	0	0	0	0

Tuesday 10 April 1

The next example shows an instance report where **Criteria** was again set to **Account**, but this time **Group by** was set to **Availability Zone**.

Criteria: Account
Group by: Availability Zone

-a list of accounts further grouped by the availability zone name

Eucalyptus Storage Report

Account	Volumes		Snapshots	
	Max (GB)	GB-days	Max (GB)	GB-days
cluster1				
eucalyptus	0	0	0	0
sales	0	0	0	0

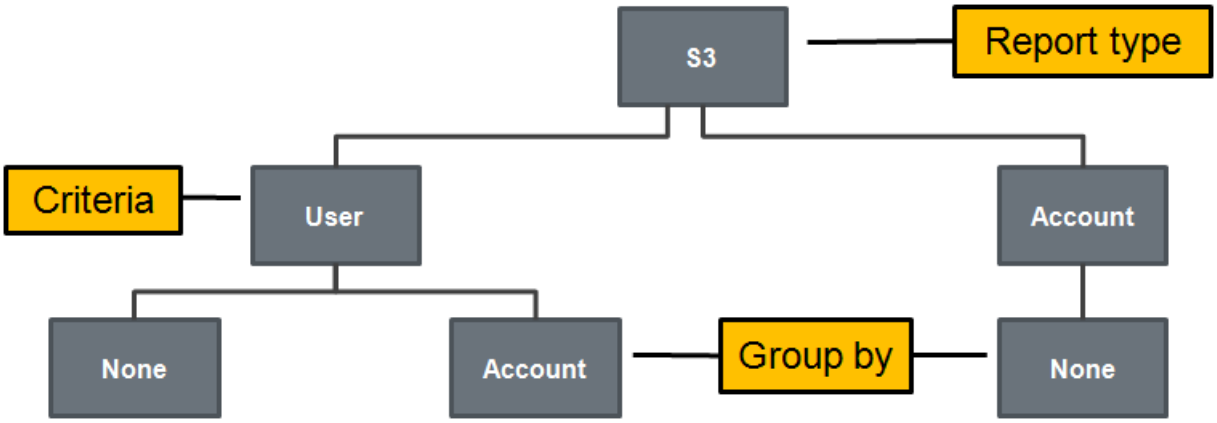
Tuesday 10 April 1

The report contains values for Max (GB) and GB-days. Max (GB) reports the maximum number of GBs that were deployed during the report period. GB-days refers to [number of days allocated] x [number of GBs allocated]. Bear in mind that number of days and GBs can be fractional, and it adds them all up. You could have 80 volumes of 100MB, each allocated for 3 hours, and the result would be 1 GB-Day (0.1*80*/(24/3)).

In these example reports, the users in the eucalyptus account consumed less than 1GB of storage controller space for less than 1 day for both volumes and snapshots. The same is true for the users in the sales account.

S3 (Walrus) Report Options

Because Walrus buckets are a cloud-wide resource and not associated with a specific Cluster Controller or availability zone, Walrus usage is reported only by user and account.



If **S3** and **User** are selected, then the **Group by** menu only has selections for **None** and **Account**.

If **S3** and **Account** are selected, then the **Group by** menu only has a selection option for **None**.

S3 Report Examples

This first example shows an S3 report where **Criteria** was set to **User**, and **Group by** was set to **None**.

Criteria: User
Group by: None

-just a list of users

Eucalyptus S3 Report

User	Buckets Max Num	Objects Max (GB)	Objects GB-days
admin	3	0	0

Tuesday 10 April 1

The next example shows an instance report where **Criteria** was again set to **User**, but this time **Group by** was set to **Account**.

Criteria: User
Group by: Account

-a list of users further grouped by account

Eucalyptus S3 Report

User	Buckets Max Num	Objects Max (GB)	Objects GB-days
eucalyptus			
admin	3	0	0

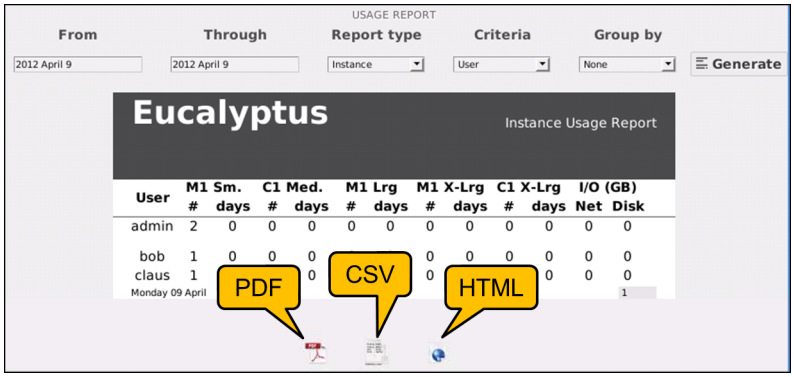
Tuesday 10 April 1

The report contains values for Max Num, Max (GB), and GB-days. Max Num reports the maximum number of buckets deployed during the report period. Max (GB) reports the maximum number of GBs that were deployed during the report period. GB-days refers to [number of days allocated] x [number of GBs allocated]. Bear in mind that number of days and GBs can be fractional, and it adds them all up. You could have 80 objects of 100MB, each allocated for 3 hours, and the result would be 1 GB-Day (0.1*80*/(24/3)).

In these example reports, the user admin in the eucalyptus account owned a maximum of 3 buckets during the report period. Those buckets consumed less than 1GB of space for less than 1 day.

Save Usage Reports

Reports can be saved to files in different formats. This makes it possible to view reports, or further process reports, using external utilities.



Note: CSV stands for *comma-separated value* and is a popular format used to input data into spreadsheet programs.

Third-Party Monitoring Tools

Eucalyptus works in conjunction with a broad ecosystem of partners. Cloud resource usage, application behavior and usage, and cloud component configuration can all be monitored through a rich set of third-party applications.

Eucalyptus HA - Introduction

Eucalyptus High Availability (HA) is a powerful technology that allows a cloud to continue operating in the event that a host running a front-end component experiences a catastrophic failure. In this section, we will cover the following topics:

- Eucalyptus HA features
- HA and redundancy
  - Service redundancy
  - Infrastructure redundancy
  - Storage redundancy
- Service states
- Arbitrator operation
- HA deployment requirements
- HA configuration

Eucalyptus HA Overview

Eucalyptus HA provides a number of capabilities, including:

- A highly-available cloud service - All cloud infrastructure management components are redundant. This includes all Eucalyptus components except for the Node Controllers.

  Note: Eucalyptus HA does **not** provide highly-available applications (instances).

- Automatic monitoring of system health, including detection of service faults - The cloud software automatically and continually monitors cloud services to determine whether they are operating correctly, or not.
- Automated failover of redundant services - Because the cloud services are continually monitored, if a service failure is detected there is an immediate failover to the redundant service.
- Administrator tools for interrogating the service health and access to service state information - Not only does the software continually monitor service states, but the administrator can also perform manual checks of service state using the Eucalyptus Administrator Console, `euca-describe-services`, and `euca-describe-arbitrators`
- Ability to remove individual component services from operation without disrupting service, assuming services are redundant - A cloud administrator can also use the Administrator Console or `euca-modify-service` to gracefully remove services from operation in order to perform maintenance on the underlying hardware or operating system. The redundant server will maintain cloud operation while the maintenance is performed on its partner service.
- Support for restoring or replacing a component service after a total-loss failure (for example, disk failure, host combustion) - Once maintenance has been completed on a machine that hosts a Eucalyptus cloud service, that machine and its cloud service can be reinserted back in to the cloud by a cloud administrator.

HA and Redundancy

One of the pillars of high availability is redundancy. If an active service or components fails, a passive service or component is brought online to maintain operation. In a Eucalyptus cloud high availability is provided by a combination of:

- Eucalyptus service redundancy
  - Cloud Controller, Walrus, Cluster Controller, Storage Controller, and VMware Broker
- Environmental redundancy

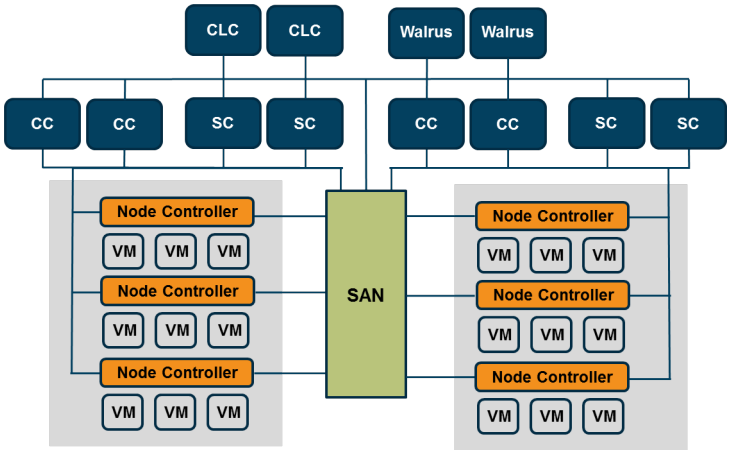
  - Underlying server and components, network, storage, datacenter power and cooling redundancy
- Operational support
  - Maintaining proper configuration and operation

It is not enough to have redundant cloud services because these cloud services still run on machines located in a datacenter. If the machines or the datacenter infrastructure fail, the cloud services can be interrupted. For this reason, the infrastructure supporting the cloud should have redundancy so that there are no, or at least very limited, single points of failure.

It is also important to develop proper operational support and procedures for running the cloud. For example, failure to properly monitor cloud services might result in failing to notice that a service has failed and is no longer redundant. If the other service were to fail, cloud operation would be interrupted. Another example would be that of not following proper administration procedures. For example, the failure to start and stop cloud services in the proper sequence might affect proper cloud operation.

Service Redundancy

Eucalyptus HA provides high availability for the cloud and cluster-layer components.



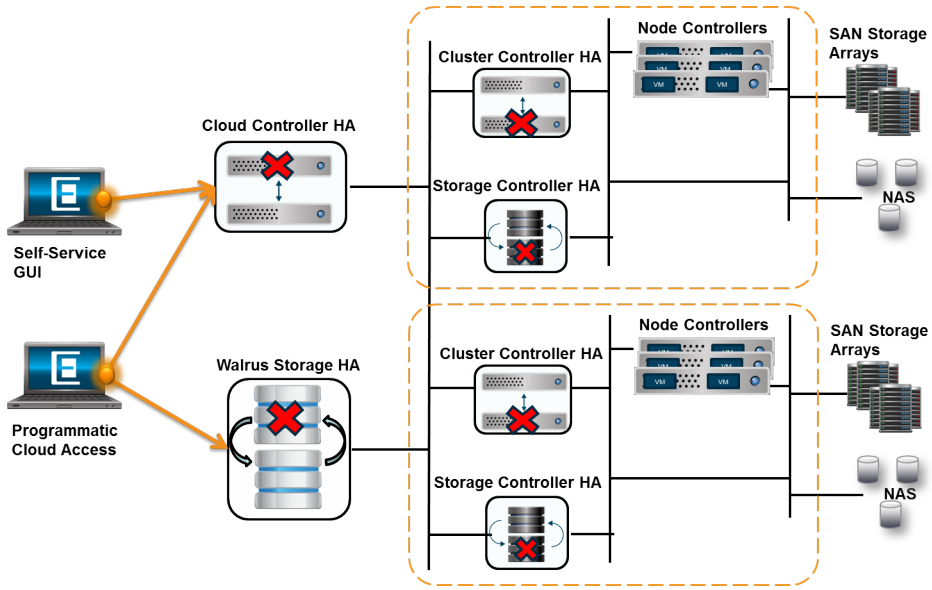
While not shown above, the VMware Broker service is also redundant. The VMware Broker service would run on the Cluster Controller hosts.

Node Controllers are not redundant within the same cluster. If a Node Controller fails, the instances that it is running will also fail. For this reason, just as Amazon recommends for its cloud, Eucalyptus also recommends that you architect your applications for availability. For example, you could ensure that a running application runs on instances in two separate clusters. If a Node Controller in one cluster fails, it would not affect instances running in another cluster.

For greater protection, you should consider redundancy in the network and storage components, as well as datacenter power and cooling components.

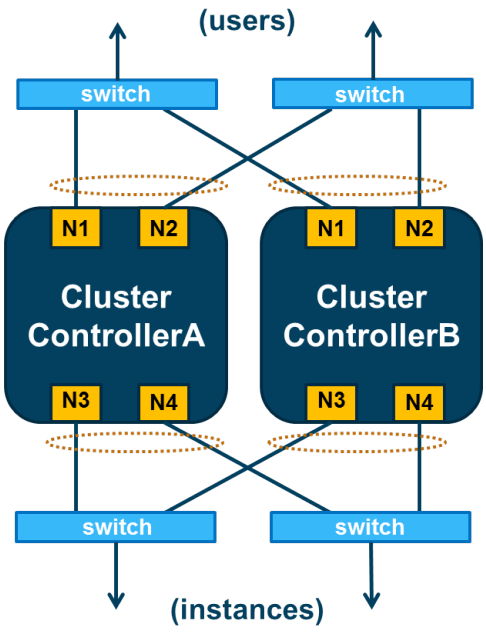
High Availability in Action

Eucalyptus HA can maintain cloud operation during multiple failures as long as both services in a service pair do not simultaneously fail and the service is available through a network interface. In the example below, six services have failed and yet the cloud would continue proper operation.



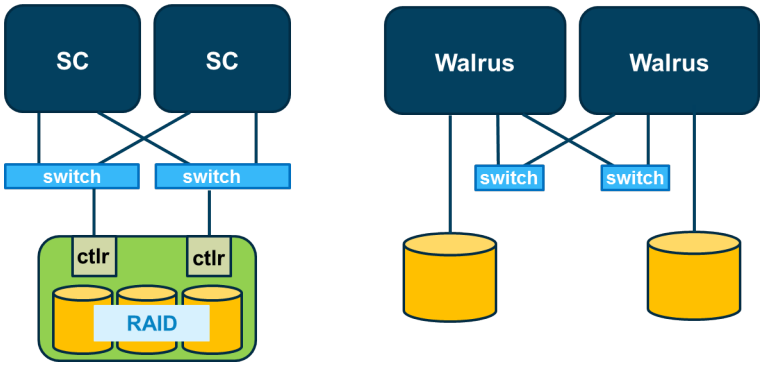
Network Redundancy

Hosts supporting Eucalyptus HA should be configured with redundant networks and components. Each host should have one extra NIC for each functional NIC. NICs should be bonded together but connected to separate physical networks. The diagram below illustrates network redundancy for the Cluster Controller. You would configure other cloud components in a similar fashion.



Storage Redundancy

Storage for the Walrus and Storage Controllers must be redundant. Consider storage path redundancy too.

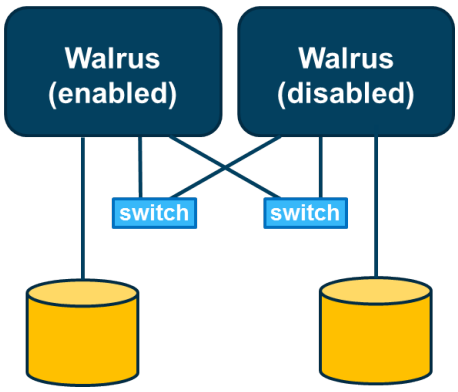


Storage Controllers require a supported SAN for Eucalyptus HA. Make sure that you have redundant paths to the storage array and that the storage array uses RAID technology to protect the data from disk failure. The storage array should also have redundant storage controllers. The point here is that you should eliminate as many single-points-of-failure as possible.

Each Walrus host requires its own storage device, however, Walrus storage is protected by Distributed Replicated Block Device (DRBD) software running in the Linux kernel.

Walrus Storage

Walrus HA requires storage disks configured in a cluster.

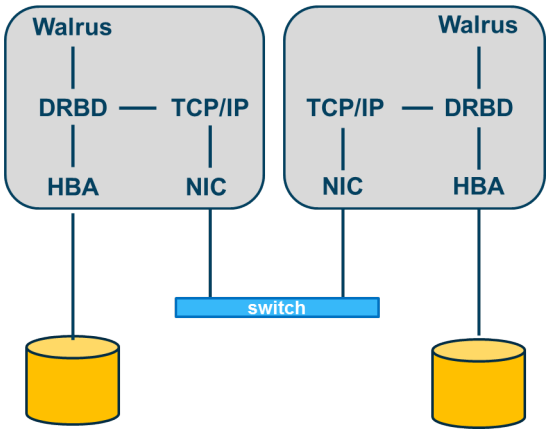


Walrus leverages kernel-based DRBD software to maintain cluster consistency. The ENABLED Walrus writes updates to storage on both Walrus hosts. At failure, the DISABLED Walrus is set to ENABLED and takes over the cluster.

Note: DRBD stands for Distributed Replicated Block Device. It is open-source software than has been present in the mainline Linux kernel since 2.6.33. It allows the configuration of a master-slave storage cluster that uses the network to maintain consistency between the two storage devices. The master-slave roles can be reversed as necessary. It allows the master to read from its own storage but send writes to both the master and slave storage. In case the master fails, the slave takes over the role of master and the storage cluster continues to operate.

DRBD Operation

DRBD is network-based mirroring (RAID 1). DRBD in the kernel intercepts disk writes and replicates them to the other Walrus using the network interface.



This image illustrates how DRBD in the kernel is able to intercept write requests to the disk and send write requests to both the local storage and to the TCP/IP driver for transport to the slave system. DRBD uses synchronous writes in Eucalyptus HA to ensure that the write is committed to the remote disk on the slave before the master Walrus flushes the data from its write buffer. This helps to ensure consistency between the clustered Walrus disks, however synchronous writes do limit the physical distance allowed between the two Walrus systems.

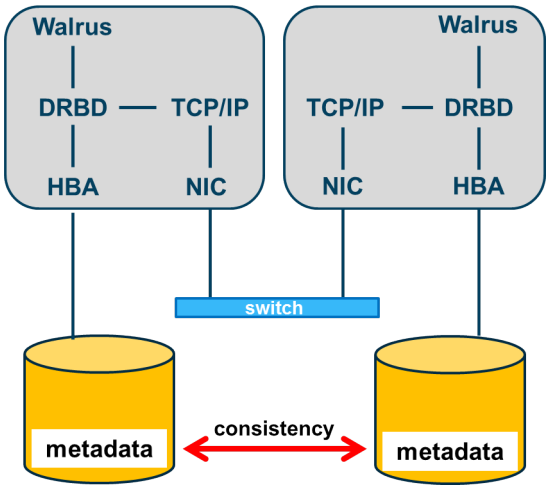
Starting with drbd-0.7-plus, up to 16 TB per DRBD device is supported.

Note: To provide clarity in the diagram, the redundant NICs and networks were removed. For true high availability protection, these would need to be factored into your HA design.

For more information, see *www.drbd.org*.

DRBD Metadata

Because DRBD is a clustered storage solution, it requires a way to maintain state information and ensure that data on both sets of storage is consistent. DRBD creates a metadata area on the Walrus disks to help ensure data consistency in the cluster. The amount of space needed to maintain this metadata is negligible. For example, metadata consumes less than 5MB even on a 150GB storage area.



Note: The online DRBD documentation at *http://www.drbd.org/docs/about/* provides a formula for calculating the size of metadata based on the size of the replicated storage.

Service States

Individual cloud and cluster services that are paired are best described as enabled and disabled. This is because one service is actively responding to requests while the other service is not. The disabled service acts as a *hot-spare* in event that the enabled service fails. A cloud administrator views service state information using the `euca-describe-services` command or the Eucalyptus Administrator Console.

These are the five possible states for services in the cloud. The normal operational states in Eucalyptus HA are ENABLED and DISABLED.

State	Operational	In Use	Description
ENABLED	yes	yes	Service is operating correctly and is selected for processing requests
DISABLED	yes	no	Service is operating correctly but is not selected for processing requests
NOTREADY	no	no	Service is failing to operate correctly
BROKEN	no	no	Service is not contactable by the system
STOPPED	n/a	no	Service has been stopped by an administrator

Cloud Availability

The cloud is available if all of the following conditions are met:

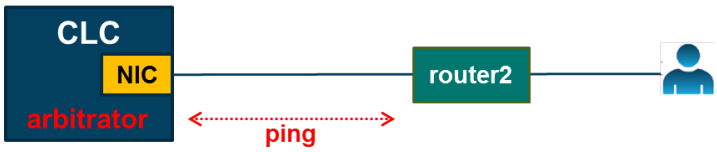
- The cloud has an enabled Cloud Controller.
- The cloud has an enabled Walrus.
- The availability zone has an enabled Cluster Controller.
- The availability zone has an enabled Storage Controller.
- VMware-hypervisor clusters have an enabled VMware Broker.
- The user-facing service has one reachable arbitrator per host (if you configure arbitrators).

Arbitrators

In Eucalyptus HA the redundant service pairs monitor each other - using a network heartbeat - to determine if they are functional or not. However, having a functional service is only useful if it can be reached by the users that need the service. To address this issue, Eucalyptus HA provides optional (but recommended) components called arbitrators that monitor network connectivity between users and public-facing cloud services. The public-facing cloud services are the Cloud Controllers, Walruses, and Cluster Controllers.

Note: Cluster Controllers are only public facing in the MANAGED and MANAGED-NOVLAN network modes.

Each arbitrator approximates network reachability by users. One or more arbitrators, running on hosts with public-facing services, use ICMP echo messages to periodically test reachability to an external IP address. These are the same ICMP messages used by the `ping` command. The external IP addresses are usually a network gateway, border router, or external site that is located on the same network that the users are located on. The logic here is that if the arbitrators running on the hosts can reach the user networks, then users should be able to reach the network where the Eucalyptus services are running. For this reason it is best to choose external IP addresses that best approximate user locations.

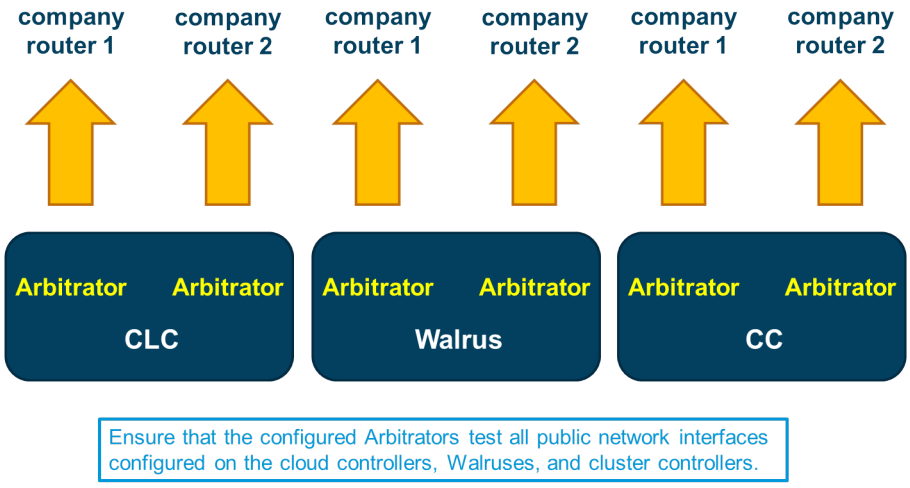


You can view arbitrator configuration and status using either the command line or the Eucalyptus Administrator Console. To view arbitrator information from the command line use one of the following commands:

```
euca-describe-arbitrators
euca-describe-properties | grep arbitrator
```

Arbitrators Example

In this example, the cloud architect determined that if the Cloud Controllers, Walrus hosts, and the Cluster Controllers could reach company routers 1 and 2, then cloud users should be able to reach back to the cloud services. Because users could be connecting from multiple networks to different network interfaces on the Eucalyptus hosts, it is recommended to configure a separate arbitrator to test network reachability through each NIC to each network where users might be. Only three arbitrators may be configured on a Cluster Controller, but more arbitrators can be configured on Cloud Controllers and Walrus hosts.



Failover from one host to another not only occurs if its Eucalyptus service fails, but also if all the arbitrators on the host fail to connect to their external IP addresses.

Eucalyptus HA Requirements

Eucalyptus HA requirements are similar to a regular Eucalyptus deployment, with the following additional considerations:

- Install each pair of services on separate hosts.
  - Cloud Controller, Walrus, Cluster Controller, Storage Controller, and if applicable VMware Broker
- The services must be installed on physical hosts.
- Service pairs must have network connectivity.
- Walrus hosts must mirror storage using DRBD.
- Storage Controller services must use a supported SAN array.

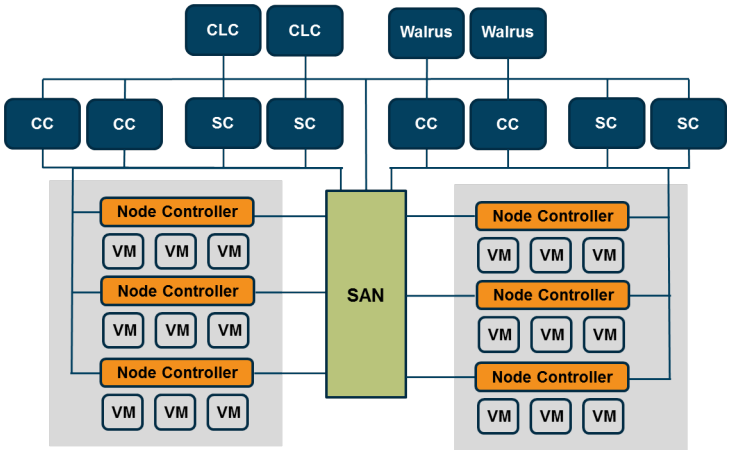
  - No iSCSI or JBOD drivers, only NetApp or EqualLogic drivers
- DNS round-robin support for Cloud Controller and Walrus hosts

Once the software has been installed on all hosts, you just register each service as you normally would. The first registered service in a pair becomes the initial primary host and is ENABLED. The second service in a pair that is registered becomes the secondary host and is DISABLED. It is DISABLED because during its initial checks it becomes aware that an ENABLED services already is running.

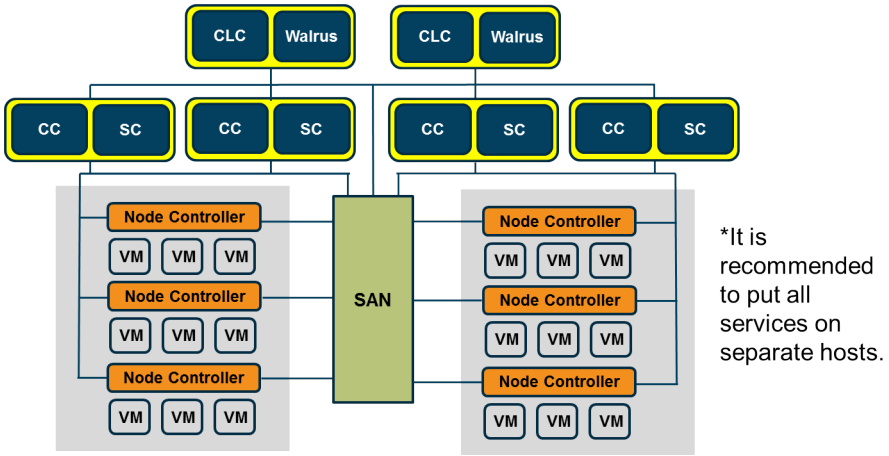
You must make sure that a firewall does not block services from each other as they use a multicast address to exchange heartbeat packets.

Example Configurations

Eucalyptus HA can be configured for full host redundancy or partial host redundant. The first illustration below is the recommended full physical host redundancy. In this configuration each separate cloud service runs on its own dedicated physical machine.



The next illustration (below) depicts partial physical host redundancy. In this example the Cloud Controller and Walrus share a physical host, as do the Cluster Controller and the Storage Controller. If any host fails and disrupts the two services running on that host, those two services are still available on the redundant host.



Install and Configure Eucalyptus HA

Eucalyptus HA installation starts with the basic Eucalyptus installation procedure. Use the same installation instructions used on the primary hosts to install the Eucalyptus software on the redundant (secondary) hosts.

You will also need to perform any pre-installation or post-installation steps on the secondary hosts that were performed on the primary hosts. This would include, for example, configuring the `/etc/eucalyptus/eucalyptus.conf` file, configuring firewalls, disabling SElinux, configuring a SAN device, and so on.

Note: Remember, Node Controllers are not redundant.

Starting the Cloud-Layer Components

Initialize only the database on the first Cloud Controller. This Cloud Controller will initially become the primary service.

```
/usr/sbin/euca_conf --initialize
```

Once the first Cloud Controller has had its database initialized, start the Cloud Controller services on both hosts, starting with the host with the initialized database.

```
service eucalyptus-cloud start
```

Start the Walrus services on both hosts.

```
service eucalyptus-cloud start
```

Note: If the Walrus is installed on the same host as the Cloud Controller, it would be started when the Cloud Controller is started and a separate command would not be necessary.

Starting the Cluster Components

Start the Cluster Controller services on both hosts.

```
service eucalyptus-cc start
```

Start the Storage Controller services on both hosts.

```
service eucalyptus-cloud start
```

Start the VMware Broker on both hosts, assuming that the cluster supports the VMware hypervisor.

```
service eucalyptus-cloud start
```

Start the Node Controllers in each cluster.

```
service eucalyptus-nc start
```

Note: While there might be multiple Node Controllers that need to be started, they are not redundant.

Repeat the startup for each cluster in the cloud.

Note: If the Storage Controller, and/or VMware Broker are installed on the same host as the Cloud Controller, they would be started when the Cloud Controller is started and a separate command would not be necessary.

Register the Secondary Cloud Controller

The first Cloud Controller is automatically registered during installation and initialization and is ENABLED. Register the second Cloud Controller with the first Cloud Controller. It will automatically start in a DISABLED state. Run the following command on the first Cloud Controller.

```
euca_conf --register-cloud --component <unique_CLC2_name> \
     --partition eucalyptus --host <CLC2_public_IP_address>
```

Note:

Because the Cloud Controller is not really part of a partition, the `--partition` option should be given the special argument `eucalyptus`. You can choose your own name for the `--component` option as long as it is unique and readily identifies the Cloud Controller for you when you see it later on in Eucalyptus Administrator Console or `euca-describe-services` command-line output. For example, the component name clc-hostB describes the service function and host location.

Register Walrus

Once you have the Cloud Controllers registered, register the first Walrus with the primary Cloud Controller. It will automatically start in an ENABLED state. Run the following command on the primary Cloud Controller.

```
euca_conf --register-walrus --component <Walrus1_unique_name> \
     --partition walrus --host <Walrus1_public_IP_address>
```

Because the Walrus hosts are not part of a cluster, the `--partition` option should be given the special argument `walrus`. You can choose your own name for the `--component` option as long as both Walrus hosts are given unique names and they readily identify the Walrus hosts for you when you see them later on in Eucalyptus Administrator Console or `euca-describe-services` command-line output. For example, the component name walrus-hostC describes the service function and host location.

Register the second Walrus with the primary Cloud Controller. It will automatically start in a DISABLED state. Run the following command on the primary Cloud Controller.

```
euca_conf --register-walrus --component <Walrus2_unique_name> \
     --partition walrus --host <Walrus2_public_IP_address>
```

Choose a unique component name for the second Walrus - for example, walrus-hostD.

Note: The partition name in both cases should be *walrus* as shown.

Register Cluster Controllers

Register the first Cluster Controller with the primary Cloud Controller. It will automatically start in an ENABLED state. Run the following command on the primary Cloud Controller.

```
euca_conf --register-cluster --component <CC1_unique_name> \
     --partition <cluster_name> --host <CC1_public_IP_address>
```

You can choose your own name for the `--component` option as long as both Cluster Controllers are given unique names and they readily identify the Cluster Controllers for you when you see them later on in Eucalyptus Administrator Console or `euca-describe-services` command-line output. For example, the component name cc-hostE describes the service function and host location..

You name your cluster when you choose the argument for the `--partition` option. You should choose a name that describes your cluster. For example, the name could describe the cluster's location, purpose, or performance characteristics.

Register the second Cluster Controller with the primary Cloud Controller. It will automatically start in a DISABLED state. Run the following command on the primary Cloud Controller.

```
euca_conf --register-cluster --component <CC2_unique_name> \
     --partition <cluster_name> --host <CC2_public_IP_address>
```

Choose a unique component name for the second Cluster Controller - for example, cc-hostF.

Note: The partition name should be the name assigned to the cluster on the primary Cluster Controller.

Repeat these steps for each cluster in the cloud.

Register VMware Brokers

Register the first VMware Broker with the primary Cloud Controller, if you are running VMware hypervisors in your cluster. It will automatically start in an ENABLED state. Run the following command on the primary Cloud Controller.

```
euca_conf --register-vmwarebroker --component <VB1_unique_name> \
     --partition <cluster_name> --host <VB1_public_IP_address>
```

You can choose your own name for the `--component` option as long as both VMware Brokers are given unique names and they readily identify the VMware Brokers for you when you see them later on in Eucalyptus Administrator Console or `euca-describe-services` command-line output. For example, the component name vb-hostE describes the service function and host location.

The name for the `--partition` option should be the same as the name of the cluster the VMware Broker will run in.

Register the second VMware Broker with the primary Cloud Controller. It will automatically start in a DISABLED state. Run the following command on the primary Cloud Controller.

```
euca_conf --register-vmwarebroker --component <VB2_unique_name> \
     --partition <cluster_name> --host <VB2_public_IP_address>
```

Choose a unique component name for the second VMware Broker - for example, vb-hostF.

Note: The partition name should be the name assigned to the cluster.

Repeat these steps for each cluster in the cloud that use VMware as the hypervisor type.

Register Storage Controllers

Register the first Storage Controller with the primary Cloud Controller. –It will automatically start in an ENABLED state. Run the following command on the primary Cloud Controller.

```
euca_conf --register-sc --component <SC1_unique_name> \
     --partition <cluster_name> --host <SC1_public_IP_address>
```

You can choose your own name for the `--component` option as long as both Storage Controllers are given unique names and they readily identify the Storage Controllers for you when you see them later on in Eucalyptus Administrator Console or `euca-describe-services` command-line output. For example, the component name sc-hostG describes the service function and host location.

The name for the `--partition` option should be the same as the name of the cluster the Storage Controllers will run in.

Register the second Storage Controller with the primary Cloud Controller. It will automatically start in a DISABLED state. Run the following command on the primary Cloud Controller.

```
euca_conf --register-sc --component <SC2_unique_name> \
     --partition <cluster_name> --host <SC2_public_IP_address>
```

Choose a unique component name for the second Storage Controller - for example, sc-hostH.

Note: The partition name should be the name assigned to the cluster.

Repeat these steps for each cluster in the cloud.

Register Node Controllers

Register the Node Controllers with the primary Cluster Controller in each cluster. On the primary Cluster Controller run the following command.

```
euca_conf --register-nodes=<“NC1_IP NC2_IP NC3_IP …”>
```

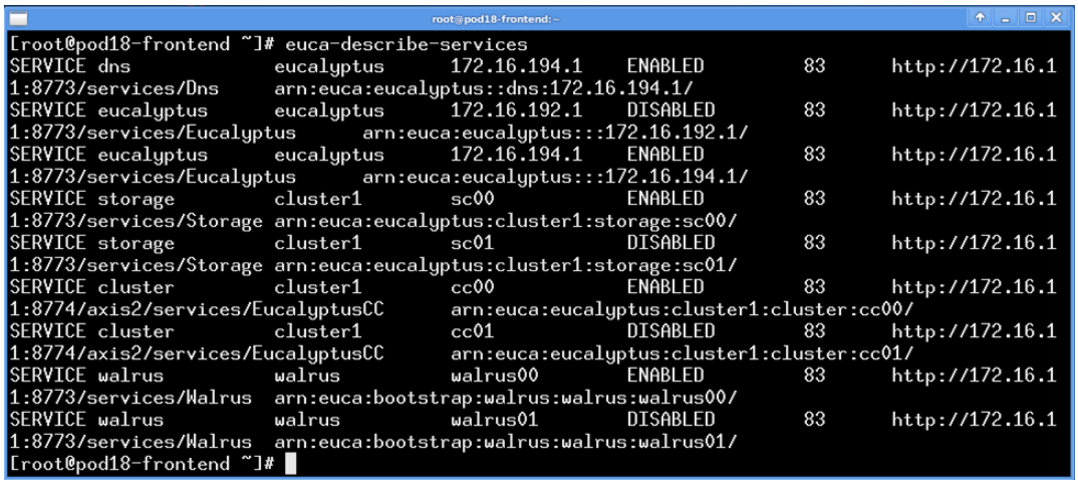
Repeat for each cluster.

Note:

While individual Node Controllers are not redundant in each cluster, there are typically multiple Node Controller hosts in each cluster.

View Service States

The `euca-describe-services` command displays service states.



The most useful information in this display are the service names, component names, and service states. For example, one of the Cloud Controller services is reported as the SERVICE *eucalyptus*, with a component name of 172.16.194.1, and is in an ENABLED state. As another example, one of the Walrus services is reported as the SERVICE *walrus*, with a component name of walrus00, and is in an ENABLED state.

It is normal for only a single DNS service to be listed. The DNS service will should be configured to automatically migrate to whichever Cloud Controller is active.

To change a service state use the syntax `euca-modify-service –s STATE component_name`. For example, `euca-modify-service –s disable walrus00`.

Service state information is also available in the Eucalyptus Administrator Console.

Register Arbitrators

Register the arbitrators on the hosts with public-facing services - the Cloud Controller, Walrus, and Cluster Controller hosts - using the following syntax:

```
euca_conf --register-arbitrator --component <arb_unique_name> \
     --partition <arb_unique_name> --host <host_public_IP_address>
```

The `--component` and `--partition` options should have the same argument. For example, you could name your arbitrators something a simple as arb00, arb01, and so on. You could also name your arbitrators after the host they run on. For example, the arbitrators on the Cloud Controllers could be name arb01-hostA and arb02-hostB.

The IP address in the `--host` option is the IP address of the NIC on the Eucalyptus host and **not** the IP address of the external network device that the arbitrator will ping.

Repeat for each arbitrator on each host.

Configure Arbitrators

Each arbitrator must be configured with an external IP address in which to send ICMP echo request messages. To configure this IP address for a Cloud Controller or Walrus use the following command.

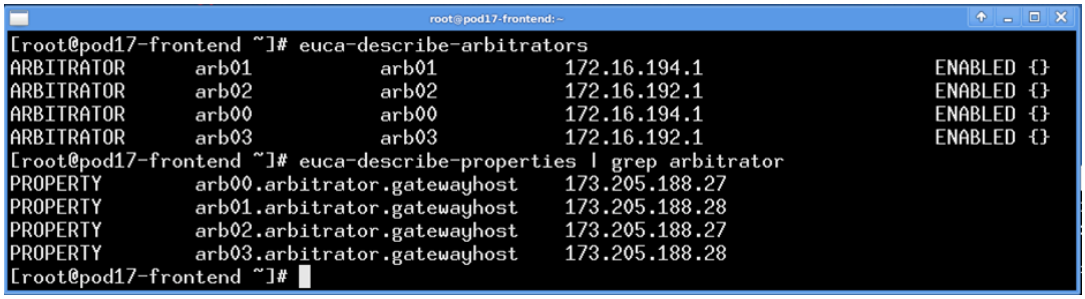
```
euca-modify-property -p    \
     <arbitrator_name>.arbitrator.gatewayhost=<address_to_ping>
```

For example, the arbitrator name could be arb00 or arb01.

To configure the external IP address for a Cluster Controller, edit its `/etc/eucalyptus/eucalyptus.conf` file and modify the `CC_ARBITRATORS=“”` parameter. You can enter up to three, space-separated IP addresses.

View Arbitrators

It takes two different command-line commands to view arbitrator configuration. The `euca-describe-arbitrators` command displays the arbitrator name, the IP address of the NIC it is associated with, and whether it is enabled or not. The `euca-describe-properties` command displays the arbitrator name and the IP address for which it is configured to send ICMP echo request messages.



In the screen capture above, the IP addresses 172.16.194.1 and 172.16.192.1 are the hosts where the arbitrators are running. Note that the arbitrators are all ENABLED. The IP addresses 173.205.188.27 and 173.205.188.28 are the IP addresses that are being used by the arbitrators to test reachability.

Arbitrator state can also be seen in the Eucalyptus Administrator Console.

Load and Use DRBD

DRBD is implemented as a loadable kernel module although it is not loaded until it is configured on the Walrus hosts. You must manually load the kernel module on each Walrus host in order to start the DRBD configuration. Enter the following command:

```
modprobe drbd
```

DRBD is used to replicate storage across two Walrus hosts, which means that Walrus must be configured to know about and use DRBD. Walrus is configured to use DRBD using parameters in the `/etc/eucalyptus/eucalyptus.conf` file and cloud database properties. In the `eucalyptus.conf file`, modify the following entry:

```
CLOUD_OPTS="-Dwalrus.storage.manager=DRBDStorageManager“
```

Then run the following commands to update cloud database properties:

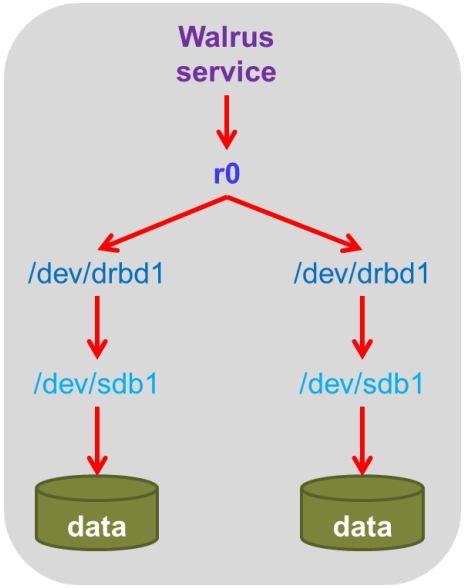
```
euca-modify-property -p walrus.blockdevice=/dev/drbd1
euca-modify-property -p walrus.resource=r0
```

The devices `/dev/drbd1` and `r0` are defined in the Eucalyptus DRBD configuration file.

Note: There is a DRBD startup script in the `/etc/init.d` directory that will start DRBD once DRBD has been configured.

Walrus and DRBD Devices

The enabled Walrus service writes to the DRBD replicated resource configured as `r0`. The `r0` resource, in turn, writes to the DRBD devices configured on the two Walrus hosts. These devices are configured as `/dev/drbd1`. The DRBD devices, in turn, write to the Linux `/dev/sdb1` devices. These Linux devices actually direct data to specific storage disks.



DRBD Configuration Files

DRBD automatically reads the master DRBD configuration file `/etc/drbd.conf` at boot time when it starts. The master DRBD configuration file allows the use of *include* statements to point to other site-specific configuration files.

The Eucalyptus site-specific DRBD configuration file is `/etc/eucalyptus/drbd.conf`. It is this site-specific file that contains all the DRBD configuration for Eucalyptus.

/etc/drbd.conf

```
include "/etc/eucalyptus/drbd.conf";
```

The Eucalyptus DRBD configuration file defines the replication information.

- Walrus hostnames
- Walrus IP addresses and ports
- Storage devices – DRBD device and backing devices
- DRBD metadata location
- Synchronization type and rate

Eucalyptus DRBD File

The file defines a replicated DRBD resource named `r0`.

Walrus writes to `/dev/drbd1` which in turn writes to `/dev/sdb1` on each Walrus host.

DRBD uses port 7789, and DRBD metadata is kept internally on `/dev/sdb1`.

This is a screen capture of the main section of the `/etc/eucalyptus/drbd.conf` file.

/etc/eucalyptus/drbd.conf

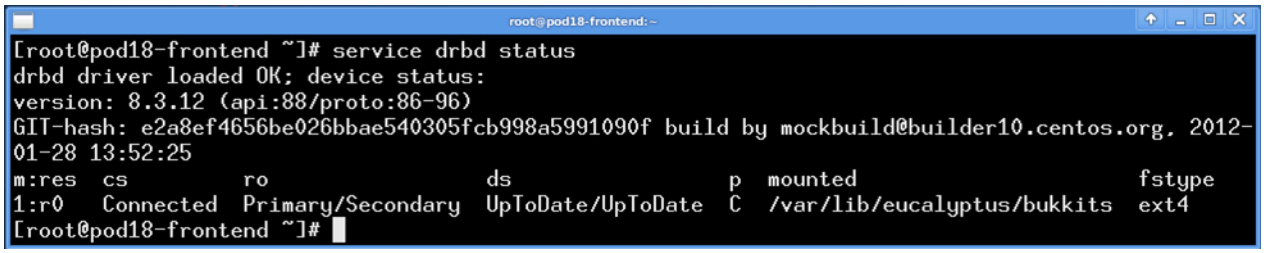
```
resource r0 {

  on pod01-frontend {
    device    /dev/drbd1;
    disk      /dev/sdb1;
    address   172.16.150.1:7789;
    meta-disk internal;
  }

  on pod02-frontend {
    device    /dev/drbd1;
    disk      /dev/sdb1;
    address   172.16.152.1:7789;
    meta-disk internal;
  }
```

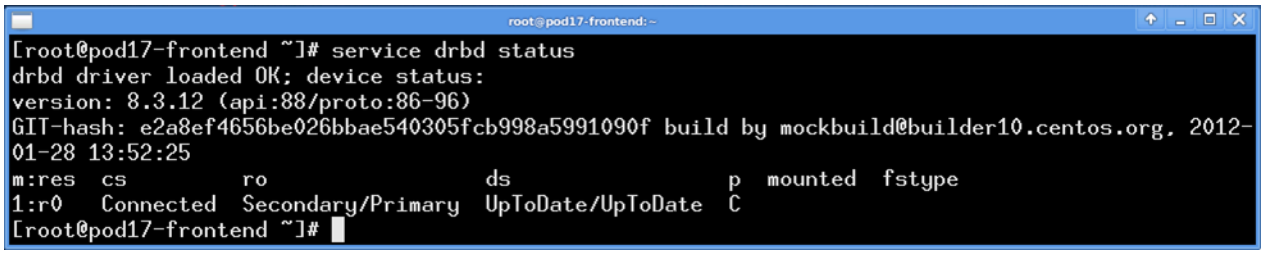
Other smaller sections of the file define the type of replication between the Walrus hosts (synchronous), the maximum replication rate (40MB/s), and other DRBD parameters.

See the online DRBD documentation at *http://www.drbd.org/docs/about/* for more information about DRBD operation, parameters, and commands.

View DRBD State

The `service drbd status` command displays DRBD state information. In the images below, its output is shown from a primary Walrus host and a secondary Walrus host.





The primary information to view in the example above is the role (ro), the connection state (cs), and the data state (ds) of the resource `r0`. The role indicates whether the host believes it is the primary or secondary DRDB server. In the example above, the first system reports itself to be the primary server and its partner to be the secondary because the display reads Primary/Secondary. The same command from the partner host confirms this because it lists itself as the secondary and its partner as the primary as seen by Secondary/Primary. The connection state of resource `r0` is Connected and both the primary and secondary servers report that the data state is UpToDate.

Also notable is that the kernel module is loaded and that the replication protocol (p) reports that the DRBD is using synchronous replication (C) between the hosts.

Note that the secondary server does not report a mounted directory or a file system type because while it is replicating the data, it is not directly interacting with Walrus.

Enable DNS Delegation

DNS delegation in Eucalyptus HA is very important. If the `eucarc` and `eucalyptus.conf` file use IP addresses instead of domain names, euca2ools commands might fail depending on which Cloud Controller is ENABLED at any given time. Modifying the `eucarc` and `eucalyptus.conf` file to use domain names instead of IP addresses will prevent this problem from occurring.

In a Eucalyptus HA configuration, there are two Cloud Controllers and the DNS service must be able to failover between them during failure events.

To enable DNS service failover:

```
euca-modify-property -p bootstrap.webservices.use_dns_delegation=true
```

For example, if the IP address of the primary and secondary Cloud Controllers are 192.168.5.1 and 192.168.5.2, and the IP addresses of primary and secondary Walrus hosts are 192.168.6.1 and 192.168.6.2, the host `eucalyptus.eucadomain.yourdomain` will resolve to 192.168.5.1 and `walrus.eucadomain.yourdomain` will resolve to 192.168.6.1.

If the primary Cloud Controller fails, the secondary Cloud Controller will become the primary and `eucalyptus.eucadomain.yourdomain` will resolve to 192.168.5.2. If the primary Walrus fails, the secondary Walrus will be promoted and `walrus.eucadomain.yourdomain` will resolve to 192.168.6.2.

Troubleshooting Eucalyptus Overview

Troubleshooting problems in a Eucalyptus cloud deployment requires in-depth knowledge of Eucalyptus and its component parts, dependencies, and configuration requirements. In this section, we will cover the following topics designed to give you a framework for troubleshooting some of the most common problems experienced in Eucalyptus cloud deployments, including:

- Troubleshooting process overview
- Troubleshooting common installation issues
- Troubleshooting instance issues
- Troubleshooting network issues
- Troubleshooting volume and snapshot issues
- Resources available to get additional assistance should these methods fail to overcome the issue you are trying to resolve

Troubleshooting Process Overview

During normal cloud operation a user submits requests to the cloud. Examples of user requests can include running an instance, creating a volume, or attaching a volume. What do you do when a working cloud is no longer properly handling user requests? How do you identify where the problem is? To resolve the problem quickly you should follow a methodical troubleshooting process:

1. Gather information about the failure
2. Check the cloud configuration
3. Check the current cloud state
4. Examine and monitor log files

Gather Failure Information

First you need to gather as much information about the failure as possible. What operation is the user trying to complete? Is the user getting an error message? Are there symptoms to report? It might be useful to know if there have been any recent changes to the cloud software, cloud hardware, or the image that the user is using.

Questions to ask include:

- What operation is failing?
- What is the error message?
- What are the symptoms?
- Has anything been recently changed?
    - Changes to the cloud configuration?
    - New software?
    - New hardware?
    - New images?

Check the Cloud Configuration Overview

To check the cloud configuration, perform the following steps:

- Gather cloud information
- Check for restrictions and misconfiguration
- Ensure multicast capabilities for Java components
- Test VLAN operation (if applicable)

- Verify EIAM policies

The steps listed above are described in more detail on the following pages.

Gather Cloud Information

It is important to know the state of the cloud configuration when troubleshooting any issue. Gather the following information:

- Configured network mode
  - MANAGED
  - MANAGED-NOVLAN
  - SYSTEM
  - STATIC
- Physical layout (topology): Where are the components located?
- Is Eucalyptus HA configured? On which components?
- Which hypervisor is in use?
- Is there a direct connect to SAN?
- Which operating systems are installed on the Eucalyptus hosts?
- Which operating systems are in use on the instance(s)?

Check For Restrictions and Misconfiguration

Look for restrictions that keep things from working. These could include cloud properties that are not set correctly.

1. For each component, check the values in `/etc/eucalyptus/eucalyptus.conf`.
2. Check for properly defined configuration parameters in the output of `euca-describe-properties`.
   - Things to look for include items like the maximum number of snapshot or maximum bucket size.

Also check configuration items that are Linux based and support, and can affect, the Eucalyptus cloud software.

1. For each component, check the network configuration – `ipconfig`, `ip addr show`, `brctl show`, `/etc/hosts`, and DNS entries.
2. Check for external firewall and Cluster Controller firewall issues.
   - Use `telnet` to test port access: `telnet <host> <port>`.
   - Run `service iptables status` on the Cluster Controller.
   - Verify that ports are opened correctly in any external firewalls.

Ensure Multicast Capabilities for Java Components

Multicast traffic needs to be allowed on switches between the Eucalyptus Java components, if the Java components are installed on separate physical hosts. Java components include the Cloud Controller, Walrus, Storage Controller, and VMware Broker. This can be tested using the following commands. One one host, run the following command to configure a multicast Java receiver:

```
java -classpath \
   /usr/share/eucalyptus/jgroups-2.11.1.Final.jar \
   org.jgroups.tests.McastReceiverTest  -mcast_addr 224.10.10.10  -port 5555
```

On the second host, run the command below to configure a Java multicast sender:

```
java -classpath \
   /usr/share/eucalyptus/jgroups-2.11.1.Final.jar \
   org.jgroups.tests.McastSenderTest  -mcast_addr 224.10.10.10  -port 5555
```

Note: The only difference in the commands above is that one is configured with `McastReceiverTest` while the other is configured with `McastSenderTest`.

Once the receiver is set up, it remains running until you exit it using a `Ctrl-C`. Each time you enter the sender command the receiver should report that it received a message.

Test VLAN Operation

If you are running in MANAGED network mode, you need to ensure that the network infrastructure connecting the Node Controllers and the Cluster Controller can properly handle VLAN tagged packets. Use the following commands to perform a test:

1. On the Cluster Controller configure a virtual network interface that is on a VLAN. In this example, it is VLAN 10:

```
vconfig add eth1 10
ifconfig eth1.10 192.168.1.1 up
```

2. On the Node Controller configure another virtual network interface that is on the same VLAN. In this example, it is VLAN 10:

```
vconfig add eth0 10
ifconfig eth0.10 192.168.1.2 up
```

3. From each host, `ping` the IP address of the other host in order to verify connectivity over the VLAN.
4. To remove the virtual network interface configuration following the test:

```
vconfig rem eth1.10
vconfig rem eth0.10
```

Verify EIAM Policies

Use either the `euare-usergetpolicy` and `euare-groupgetpolicy` commands or the Eucalyptus Dashboard to look at EIAM policies for a user or group to determine if the user is not allowed to perform the action. The command line syntax is as follows:

```
euare-groupgetpolicy –g <group> -p <policy>
euare-usergetpolicy –u <user> -p <policy>
```

You must understand how to evaluate multiple conflicting policies if they exist. There is no way in the current Eucalyptus Dashboard or command line to evaluate layered policies to see the combinatory effects. You must manually evaluate the policies.

Check the Current Cloud State

The following steps can be used to discover information about the current state of the cloud:

1. Ensure that cloud services are registered properly with `euca_conf --list-<component>`.
   - Use `euca_conf –-list-*` to confirm that components are registered properly.
   - Ensure that services are registered at the proper IP address.
2. Check `euca-describe-services –E` for any failures. It will list IP addresses, component names, partition names, and statuses, and provides you with the last known failure that caused a service transition from an operational state to an error state.

   Note: You can filter the results by service state by adding the following option: `-F <ENABLED|DISABLED|NOTREADY>`. `ENABLED` will show you what the cloud sees as ready to accept and process requests.

   Note: The Node Controllers will not be displayed in the output of `euca-describe-services`.

3. Check instance resource availability:
   - CPU resources on Node Controllers can be checked by using `euca-describe-availability-zones verbose`.

- Public IP addresses can be checked with `euca-describe-addresses verbose`.
- Storage resources on the Node Controllers can be checked by running `df -h` on each Node Controller in the cluster. The command `euca-describe-availability-zones verbose` will tell you whether or not the cloud has sufficient storage resources to launch another instance.
- Memory resources on the Node Controllers can be checked by running `cat /proc/meminfo` on each Node Controller in the cluster. The command `euca-describe-availability-zones verbose` will tell you whether or not the cloud has sufficient memory resources to launch another instance.

Note: For an instance to launch, there must be sufficient available CPU, RAM, and disk resources on at least one Node Controller to support it.

4. Check available disk space on the Storage Controller and Walrus using `df –h`.
5. To gather and view a large amount of configuration information, use:

```
for cmd in `ls -l /usr/bin/euca-describe-*`;  \
  do $cmd; done | less
```

Ensure that NTP is enabled and time is synchronized across all Eucalyptus hosts using `ps ax | grep ntpd`, `date`.

Examine and Monitor Log Files

Eucalyptus logs are located on each Eucalyptus host at `/var/log/eucalyptus`.

Java components (Cloud Controller, Storage Controller, and Walrus) log files include:

- `cloud-output.log`
- `cloud-error.log`

Other log files exist for these components, but these are typically the most useful. Be aware that logs are regularly rotated so you will see archived copies of logs too.

The VMware Broker produces an additional file which records the process of converting image files to VMDK format for upload to ESX/ESXi hosts.

- `euca_imager.log`

The C-based components (Cluster Controller and Node Controller) log files include:

- `cc.log`
- `nc.log`

Log files can be opened and read using any standard text viewing tool, including `less`, `nano`, and `vi`.

Log files can be monitored in real time from the command line using the `tail -F <log file>` command. To stop monitoring, press `Ctrl-C`.

Common Installation Issues

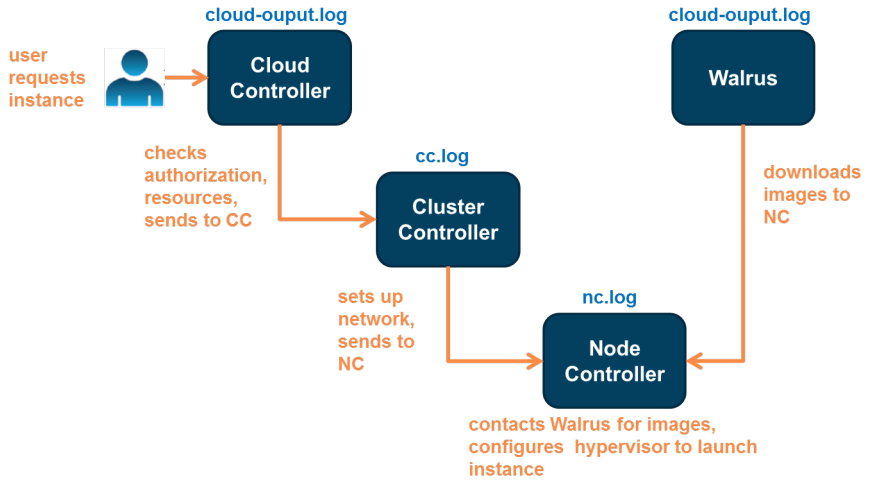
A proper Eucalyptus installation requires numerous command-line commands and text file edits. Proper configuration requires great attention to detail, and sometimes even careful individuals can make a typographical error that causes the cloud not to function properly. Some of the more common installation issues to check are:

- Syntax problems in `/etc/eucalyptus/eucalyptus.conf`
  - Commented lines, missing commas, comma instead of a period, missing double quotation mark, and so on.
- Misconfiguration of `/etc/hosts`
  - Syntax should be: `<IP_address> <host.domainname> <host>`
- Node Controller or other service IP address incorrectly registered
- Mismatch between Cluster Controller and Storage Controller partition name in a cluster

Troubleshoot Instance Issues

Instance problems generally fall into three categories: resource availability issues, image problems, and cloud component issues.

It is useful to understand the launch process before trying to troubleshoot instance problems. The following diagram illustrates what happens when a new instance launches:



Understanding this process allows you to trace problems across the components. For example, you can check the Cloud Controller log to see if it sent messages to the Cluster Controller. Similarly, you can check the Cluster Controller log to see if it sent messages to the Node Controller. Did the Walrus send images to the Node Controller? Did the Node Controller send the instance command to the hypervisor?

When troubleshooting, it is usually recommended to flip the order around and start at the Node Controller. Examine its log files, then work your way back to the Cluster Controller and finally the Cloud Controller until the issue is identified.

Instance Resource Availability

If you cannot launch or connect to an instance, before you troubleshoot further you should first make sure that the problem is an actual error condition as opposed to a lack of available resources. Check the cloud state to ensure you have enough CPU, memory, disk, and public IP addresses available for the instance to rule out a resource-related issue. The command `euca-describe-availability-zone verbose` and `euca-describe-addresses verbose` are useful to verify resource availability.

Image Problems

The image you are trying to boot from might have an issue. In some cases the instance will not launch. In other cases, the instance will launch, go to a pending state for some time, and then shut down. This could indicate either an error in the cloud services or an error in the image itself.

Another good instance troubleshooting step is to boot from a known good image. If this second image boots, this is a strong indication that the first image has a problem.

You can also attach to the console of an instance during the boot process using the `euca-get-console-output` command. This allows you to view the boot messages or see if the instance reaches a login prompt. The boot messages might provide an indication as to why the image is not booting.

Cloud Problem: Node Controller

If you suspect a problem with the cloud is causing an instance launch problem, start your troubleshooting efforts at the Node Controller.

1. To find the Node Controller where the instance is running or trying to run, use `euca-describe-nodes` which lists the Node Controllers and the instance IDs running on each of them. You must test the problem on the same Node Controller that the user experienced the problem.
2. Use the `grep` utility to search `nc.log` file on the Node Controller for the relevant instance ID:

   ```
   cat /var/log/eucalyptus/nc.log | grep <instance_ID>
   ```

   If image has been handed to the hypervisor, it is likely an image problem and not cloud problem.
3. Look for errors in downloading items from Walrus.
4. Look for `libvirt` errors in launching an instance.

If no instance data is found in the `nc.log` file on the Node Controller, then it is likely that the problem could be listed in the Cluster Controller log file. Move to that host to continue troubleshooting.

Cloud Problem: Cluster Controller

If the Node Controller does not report the instance, check the Cluster Controller to ensure that it reports sending the instance to the Node Controller. Search the `cc.log` file for `RunInstances` using the following command:

```
cat /var/log/eucalyptus/cc.log | grep RunInstances [| less]
```

Note: The last part of the command (`| less`) is optional, but may be useful if a large amount of data is returned. You can also use `grep` to search for the instance ID.

Note: You can also use `grep` to search for instance ID in the `cc.log` on the Cluster Controller. This will display log information related to launching and monitoring the instance.

If the command was issued by the Cluster Controller but not received by the Node Controller, this could indicate a communication problem between the two hosts.

If there is no instance data in the `cc.log` file, then it is likely that the problem could be listed in the Cloud Controller log file. Move to that host to continue troubleshooting.

Cloud Problem: Cloud Controller

If neither the Node Controller nor the Cluster Controller report the instance, check the Cloud Controller and ensure that it reports sending the instance to the Cluster Controller. Search the `cloud-output.log` file for `euca:RunInstancesType` using the following command:

```
cat /var/log/eucalyptus/cloud-output.log | grep euca:RunInstancesType \
   [| less]
```

Note: The last part of the command (`| less`) is optional, but may be useful if a large amount of data is returned.

Note: You can also use `grep` to search for the instance ID in the `cloud-output.log` file on the Cloud Controller.

Troubleshoot Network Issues

Both the Cluster Controller and the Node Controller are involved in the network operation of the instance.

In a MANAGED network mode configuration, make sure that the minimum and maximum VLAN IDs are set properly, and that switches are forwarding packets to these VLANs. To view the VLAN ID range used `euca-describe-properties` and look for the following entries:

- `cloud.network.global_max_network_tag`
- `cloud.network.global_min_network_tag`

If your networking environment is already using VLANs for other reasons, Eucalyptus supports the definition of a smaller range of VLANs. To set this range with a running and configured Eucalyptus installation:

```
euca-modify-property -p cloud.network.global_max_network_tag=<max_vlan_tag>
euca-modify-property -p cloud.network.global_min_network_tag=<min_vlan_tag>
```

If this fails, use traditional network and operating system troubleshooting on the Eucalyptus hosts in order to make sure that networking between the hosts is working properly.

Firewall Issues

If you suspect a firewall issue, go through the following procedures:

- Run the `euca-describe-groups verbose` command, which describes the access that the outside world has to instances in the cloud. Make sure that `ping` and `ssh` are allowed.
- Ensure the Cloud Controller `cloud-output.log` file has a message with `euca:RunInstancesType` that contains the correct network rules and IP addresses.
- Verify rules in `iptables` on the Cloud Controller using the following command: `service iptables status`.

Troubleshoot Volume and Snapshot Issues

Snapshot and volume issues generally manifest as attachment or creation issues.

Volume Attachment Issues

The basic steps in looking for clues to troubleshoot volume attachment issues are:

1. Locate the relevant Node Controller.
2. Search the `nc.log` file for the relevant volume ID.
3. Interrogate iSCSI targets on the SAN or Storage Controller.
4. Ensure that `connect_iscsi_target` was called.

Look for errors as to why volume attachment failed. The `euca-describe-nodes` command displays the instance IDs that are running on each Node Controller. Once you locate the Node Controller that experienced the problem, use the following command to search for the volume ID that failed on that host to look for clues as to why it failed to attach:

```
cat /var/log/eucalyptus/nc.log | grep <volume_ID>
```

You can also interrogate the iSCSI target on the Storage Controller using the `tgtadm` command. The `tgtadm` command should show a target/LUN connected to the Node Controller. The Storage Controller creates a target for each volume, and each target includes a LUN 0 controller and a LUN 1 disk, which is the size of the volume. In the output of the command, look for the following section beneath the Target number:

```
I_T nexus information:
  I_T nexus: 3
    Initiator: <Node_Controller_iSCSI_initiator_IQN>
    Connection: 0
       IP Address: <Node Controller IP Address>
```

Note: If you are using a supported iSCSI SAN array to create volumes, use the array management software to interrogate for iSCSI targets.

Searching for `connect_iscsi_target` in the `nc.log` file will provide information about the device name (`/dev/sdN`) that was provided to the instance to access the volume. If this is located, look for any issues in attachment such as a lack of loop devices.

Volume Creation Issues

When troubleshooting volume creation issues, ensure that volume configuration parameters are set properly for the requested resource size. Also check that user is not trying to exceed some cloud defined maximum. Run the `euca-describe-properties` command and verify the total-volume or per-volume size is not exceeded. Look for the following output:

- `<cluster_name>.storage.maxtotalvolumesizeingb`
- `<cluster_name>.storage.maxvolumesizeingb`

Quotas can also cause volume creation problems. Check for EIAM quotas that might have the following quota keys:

- `ec2:quota-volumenumber`
- `ec2:quota-volumetotalsize`

Snapshot Issues

When experiencing problems creating a snapshot from a volume:

1. Check that volume has not been deleted by running `euca-describe-volumes`.
2. Ensure that Walrus has enough space for snapshot using `df –h`.
3. Check that system properties or quotas have not been exceeded using `euca-describe-properties` (`walrus.storagemaxtotalsnapshotsizeingb`) and by checking EIAM policies for the following quota key (`ec2:quota-snapshotnumber`).

   Note: You might want to check both maximum size and maximum total size parameters, if applicable.

4. Check the log files on the Node Controller (`nc.log`) and on the Storage Controller and Walrus (`cloud-output.log`).

Additional Resources

You may find the following resources helpful when troubleshooting unusual, operating-system-specific or hardware-specific issues:

- Support Center articles at https://engage.eucalyptus.com/
- The **Troubleshooting Eucalyptus** section of the Eucalyptus Administration Guide
- Eucalyptus documentation at http://www.eucalyptus.com/eucalyptus-cloud/documentation
- Searching for error messages using a Web search engine
- Amazon Web Services documentation at http://aws.amazon.com/documentation

Optional Lab - Troubleshoot an Instance Launch Failure

In this lab exercise you will walk through the process of troubleshooting an instance launch failure, had one occurred. We will assume this cloud has been functioning normally and that this user has appropriate permissions to launch an instance and has done so in the past.

Lab Objectives:

- Run commands to gather information
- Examine log files

Gather data

In this section of the lab you will examine the state of your cloud and gather information to use when examining log files. While you already know the results that some of these commands will produce, you should pretend the cloud you are using is unfamiliar to you and walk through the entire process.

1. Desktop From the Debian desktop, if necessary use SSH to log in to the front-end host.

```
# ssh <front_end_public_IP>
```

2. Front End Launch a new instance. Wait for it to enter a running state.

```
# euca-describe-images
# euca-run-instances –k <keypair_name> emi-<nnnnnnnn>
# euca-describe-instances
```

Note: You should have at least one running instance for this lab.

3. Front End Check to make sure that the Eucalyptus components are still registered correctly.

```
# euca_conf --list-clusters
# euca_conf --list-scs
# euca_conf --list-walruses
# euca_conf --list-nodes
```

4. Front End (Optional) Examine the output of all `euca-describe-*` commands in the `/usr/sbin` directory.

```
# find /usr/sbin –name ‘euca-describe-*’ –exec ‘{}’ \;
```

5. Front End Check for events in any of the Eucalyptus cloud services.

```
# euca-describe-services -E
```

6. Front End Check to make sure all front-end components are enabled. This command filters the output and shows only those services with an enabled state. You could also filter for DISABLED, NOTREADY, BROKEN, or STOPPED.

```
# euca-describe-services -F ENABLED
```

7. Front End Verify that Network Time Protocol (NTP) is running and that time is synchronized on every host running a Eucalyptus service.

```
# pgrep ntpd
# ntpq -p          (see note below)
```

Note: For correct synchronization, the delay and offset values should be non-zero and the jitter value should be under 100.

8. Front End Check for available CPU, memory, and storage resources.

```
# euca-describe-availability-zones verbose
```

9. Front End Check available disk space on the Node Controller, Storage Controller, and Walrus.

```
# df -h
```

10. Front End Check cloud configuration parameters.

```
# euca-describe-properties
```

11. Front End Find the instance ID number of the *failed* instance.

```
# euca-describe-instances
```

Note: For lab purposes, pick any instance ID number in the list as the instance that you will consider as the *failed* instance in this lab, and write down its ID number.

12. Front End Find the IP address of the Node Controller to which this instance was assigned.

```
# euca-describe-nodes
```

Examine log files

In this section of the lab you will examine log files of the Node Controller, Cluster Controller, and Cloud Controller for information regarding the *failed* instance.

1. Desktop From the Debian desktop, use SSH to log in to the Node Controller to which the *failed* instance was assigned.

```
# ssh <node_public_IP_address>
```

2. Node Search the `/var/log/eucalyptus/nc.log` file for information about the *failed* instance.

```
# grep <instance_ID> /var/log/eucalyptus/nc.log | less
```

Examine:

- Did the Node Controller get the *doRunInstance* command?
- Did it have the right vmtype parameters (cores, memory, storage.)?
- Did it allocate artifacts for ephemeral space and swap space?
- Did the Node Controller download EMI, EKI, or ERI files from the Walrus?
- Did Walrus grant the requests to access images (correct bucket permissions)?
- Did the Node Controller try to boot the instance?
- Did the following state changes occur?
  - Staging > Pending (booting)
  - Were IP, MAC, VLAN, and network assigned
  - Were platform and volume assigned?

- Block device stats initialized - block devices > blk bytes (0), net devices > net bytes (0)
- Pending > Running (Extant)

3. Node Assume that in your `nc.log` investigation you discovered that the *doRunInstance* command was not received from the Cluster Controller.

4. Desktop From your Debian desktop, open another xterm window, use SSH to log in to the Cluster Controller host (the front-end host).

```
# ssh <front_end_public_IP>
```

5. Front End On the front-end host, perform a similar investigation on its `/var/log/eucalyptus/cc.log` file.

```
# grep <instance_ID> /var/log/eucalyptus/cc.log | less
```

Did the instance get to the Running (Extant) state according to the Cluster Controller?

6. Front End Assume that in your `cc.log` investigation the instance ID number does not appear. The next place to check would be the `/var/log/eucalyptus/cloud-output.log` file on the Cloud Controller. Since you are running the Cluster Controller and Cloud Controller on the same physical host, you do not have to SSH to the Cloud Controller in order to examine the log file - you are already logged on to the Cloud Controller host. in your second xterm window.

```
  # grep <instance_ID> /var/log/eucalyptus/cloud-output.log | less
```

- Did the *euca:RunInstancesType* request get logged?
- Are the *NetworkRule*: settings correct?
- Search for errors in the less output:

```
  /ERROR         (press n to move through the list)
```

Note: For all log examinations, if the instance problem occurred in the more-than-recent past, data about the failure might be in a log file that has been archived and renamed (`*. log.1`, `*. log.2`, and so on.)

Note: To examine all available log files and not just the current log files, append the file name with an asterisk. For example, replace `cloud-output.log` with `cloud-output.log*` in the `grep` command.