Department of Computer Science and Engineering
National Institute Of Technology Calicut

ZEUS Cloud
User's Manual

http://zeus.nitc.ac.in

Supervisor: Dr. Vineeth Paleri
Faculty-In-Charge: Dr. Abdul Nazeer

Contents

Chapter 1

Introduction

ZEUS Cloud is a private cloud that has been set up within the institute to be used by the students for research or project work. The cloud is powered by the popular cloud computing software Eucalyptus. ZEUS Cloud is capable of virtualization of machine images and high performance computing. The cloud is scalable, in terms of adding more resources to the existing system and allocation of resources to users according to their needs. Users will be able to try and test platforms and packages required for their projects, and develop, test and debug their programs. The goal of ZEUS Cloud is to deploy computational resources through Local Area Network within the institute for student projects and research work.

1.1 Features

- **Virtualization** : Users can prepare VMs for use in the cloud according to their requirements (software or hardware). Images needs to be bundled, uploaded and registered with the cloud. Users can deploy several instances of images available in the cloud for their use, and every user will have full privileges on the VMs.
- **High Performance Computing** : Users can deploy multiple instances of same image on the cloud. A shell script has been provided which will setup MPI (Message Passing Interface for Parallel Computing) in multiple machine instances (Ubuntu), so that running MPI applications is very convenient.

Chapter 2

Getting Started

2.1 Cloud Registration

The cloud can be accessed by a web-interface provided by Eucalyptus. Before you can access the cloud, you must first sign up for an account using the web interface. To access the web interface you must be within the Local Area Network (LAN) of the institute.

1. Open your browser and goto https://zeus.nitc.ac.in:8443/.
2. Click Apply to access the application form.
3. Fill out the Eucalyptus account application form with your institute email address (@nitc.ac.in).

The administrator will review your application and approve your request depending on your application. The more complete the information you provide on your account application the easier for the administrator to verify your identity. Once the administrator approves your request, your account is created.

2.2 User Credentials

You must have proper credentials to use client tools (such as Euca2ools) from your PC to interact with ZEUS Cloud. To get user credentials

1. Log in to the web interface using your username and password.
2. Click your username at the top of the screen and then click Download new credentials. The download contains a zip-file with your public/private key pair, a bash script, and several other required files.
3. Unzip your credentials zip file to a directory of your choice and change permissions as shown.

```
mkdir ~/.euca
cd ~/.euca
unzip <filepath>/euca2-<user>-x509.zip
chmod 0700 ~/.euca
chmod 0600 *
```

Chapter 3

Euca2ools

3.1 Introduction

Euca2ools are command-line tools for interacting with ZEUS Cloud. Euca2ools helps in querying resource availability, key management, VM management, Image management, security group and volume management.

3.2 Installing Euca2ools

On Ubuntu based operating systems the following commaxnd installs Euca2ools and its dependancies.

```
apt-get install euca2ools
```

For installation on other operating systems refer Eucalyptus Wiki (`http://open.eucalyptus.com/wiki`).

3.3 Using Euca2ools

Before using Euca2ools ensure that the environment variables are set by sourcing the eucarc file.

```
. ~/.euca/eucarc
```

3.3.1 Querying the System

The following command shows the availability zones i.e. the resources available to access. 'verbose' added at the end shows the number of available/total machines in the zone.

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

The following command shows all the machine images available for a particular user.

```
euca-describe-images
```

There are three types of images :

- OS image <emi-XXXXXXXX> : Contains the actual image of the Operating system.(*.img file)
- Kernel image <eki-XXXXXXXX> : This is the kernel image for an OS. It can be bundled to an OS Image or specified separately while creating an instance. (vmlinuz* file)
- Ramdisk image <eri-XXXXXXXX> : This is the ramdisk image used to run an OS. Most often this is optional. (initrd* file)

The following command shows all the operating system instances created and running by a particular user. It shows the details of the instances like instance ID (i-XXXXXXXX), Image ID, machine type (small, medium, large etc.), current status (pending, running, terminated etc.) etc.

```
euca-describe-instances
```

3.3.2 Keypair Management

Keypairs are used to authenticate a user's identity. Before running a VM instance, you must first create a keypair as follows:

```
euca-add-keypair mykey | tee mykey.private
```

A pair of keys are created; one public key, stored in the cloud, and one private key stored in the file mykey.private and printed to standard output. The ssh client requires strict permissions on private keys:

```
chmod 0600 mykey.private
```

All keypairs associated with a user can be seen by the command:

```
euca-describe-keypairs
```

3.3.3 Image Management

Bundling and Uploading Images

A user can bundle and upload Eucalyptus compatible Operating system images which can be used to create instances. The images are uploaded to the walrus storage controller located at the central eucalyptus server.

- Kernel file (vmlinuz* file) :

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

  Here <kernel-file> is the vmlinuz* file and <kernel-bucket> is the user specified name to denote this file. euca-register will return an ID (<eki-XXXXXXXX>).

- Ramdisk file (initrd* file) :

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

Here <ramdisk-file> is the initrd* file and <ramdisk-bucket> is the user specified name. euca-register will return an ID (<eri-XXXXXXXX>).

- Image file (*.img file) :

```
euca-bundle-image -i <image-file> --kernel <eki-XXXXXXXX> \
--ramdisk <eri-XXXXXXXX>
euca-upload-bundle -b <image-bucket> -m /tmp/<image-file>.manifest.xml
euca-register <image-bucket>/<image-file>.manifest.xml
```

Here <image-file> is the *.img file and <image-bucket> is the user specified name. euca-register will return an ID (<emi-XXXXXXXX>).

Deleting Images

To delete an Image, first it has to be deregistered

```
euca-deregister <emi-XXXXXXXX>
```

To remove the image and bucket, use the following code:

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

3.4 Creating and Running Virtual Instances

Before running an instance, Euca2ools must be authorized to allow 'ssh' connections from the Internet:

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

To run an instance of an OS, use the following code

```
euca-run-instance -k <keypair> -t <machine-size> -n <number-of-instances> \
  <emi-XXXXXXXX>
```

Here <keypair> is the keypair used to login to an instance (euca-describe-keypairs), <machine-size> is the size of the machine required (m1.small, c1.medium, m1.large etc.) and <emi-XXXXXXXX> is the Image ID of the OS to be run (euca-describe-images).

Once an instance is created the status of the instance is 'pending' (euca-describe-instances). Once it changes to 'running', the instance can be accessed by ssh.

```
ssh -i <private-key> <user-name>@<accessible-ip-address>
```

Here <private-key> is the mykey.private file created, <user-name> is the login username of the machine (eg. ubuntu, root etc.) and <accessible-ip-address> is the IP address by which the instance can be accessed (euca-describe-instances).

To reboot the instance:

```
euca-reboot-instances <instance-ID>
```

To terminate the instance

```
euca-terminate-instance <instance-ID>
```

Here <instance-ID> is the ID of the running instance (euca-describe-instances).

Chapter 4

Running MPI Programs

4.1 Introduction

MPI (Message Passing Interface) is a language-independent communications protocol used to program parallel computers. Both point-to-point and collective communication are supported. The MPI interface is meant to provide synchronization, and communication functionality between a set of processes. These processes can run on different nodes/systems/instances. MPI programs are created to run parallel programming algorithms.

4.2 Setting up MPI

Create and run the required number of instances using a single keypair. It is assumed that the instances are of Ubuntu.

```
euca-run-instance -k <keypair> -n <number-of-instances> \
 <emi-XXXXXXXX>
```

Copy the following script to a file eg. mpi-setup.sh

```
#! /bin/bash
# . mpi-setup.sh <keypair> <no-of-nodes> "node-ip1
   node-ip2 ..."

KEY=$1
NUMNODES=$2
NODES=$3
IFS=' '
read -ra ARRAY <<< "$NODES"

for i in "${ARRAY[@]}"
do
        eval `scp -i $KEY -o StrictHostKeychecking=no
           $KEY ubuntu@$i:~/`
done

for i in "${ARRAY[@]}"
do
        ssh -i $KEY -o StrictHostKeychecking=no
           ubuntu@$i 'ssh-keygen -t rsa -N "" -f /home
           /ubuntu/.ssh/id_rsa'
done

touch auth_keys
ssh -i $KEY -o StrictHostKeychecking=no ubuntu@${ARRAY
   [0]} 'cat /home/ubuntu/.ssh/authorized_keys' >>
   auth_keys

for i in "${ARRAY[@]}"
do
        ssh -i $KEY -o StrictHostKeychecking=no
           ubuntu@$i 'cat /home/ubuntu/.ssh/id_rsa.pub
           ' >>auth_keys
done

for i in "${ARRAY[@]}"
do
        eval `scp -i $KEY -o StrictHostKeychecking=no
           auth_keys ubuntu@$i:~/.ssh/authorized_keys`
done

rm auth_keys

for i in "${ARRAY[@]}"
do
        ssh -i $KEY -o StrictHostKeychecking=no
           ubuntu@$i 'sudo apt-get install openmpi-bin
            openmpi-doc libopenmpi-dev'
done
```

Run the script as:

```
. mpi-setup.sh <keypair-location> <no-of-nodes> "node-ips"
```

where <keypair-location> is the location of the private key and "node-ips" are the IP addresses of the instances each separated by a space.

4.3 Creating an MPI Program

After running the mpi-setup.sh script, ssh into an instance:

```
ssh -i <private-key> ubuntu@<accessible-ip-address>
```

In the instance, create a C file with the MPI program in it. Here is a sample HelloWorld program using MPI (hello.c).

```
/* C Example */
#include <stdio.h>
#include <mpi.h>


int main (argc, argv)
     int argc;
     char *argv[];
{
  int rank, size, len;
    char name[MPI_MAX_PROCESSOR_NAME];


  MPI_Init (&argc, &argv);      /* starts MPI */
  MPI_Comm_rank (MPI_COMM_WORLD, &rank);
                                /* get current process
                                    id */
  MPI_Comm_size (MPI_COMM_WORLD, &size);
                                /* get number of
                                   processes */
  printf( "Hello world from process %d of %d from %s\n
     ", rank, size,name );
  MPI_Finalize();
  return 0;
}
```

4.4 Running an MPI Program

Create a machinefile which contains the IP addresses of the instances where MPI should be run. Here is a sample machinefile (machine.txt):

```
192.168.41.210
192.168.41.211
192.168.41.212
...
```

Complile the C program using the MPI command mpicc:

```
mpicc -o hello.o hello.c
```

Copy the output binary (hello.o) to all the other instances:

```
scp hello.o ubuntu@192.168.41.210:~/
scp hello.o ubuntu@192.168.41.211:~/
scp hello.o ubuntu@192.168.41.212:~/
...
```

Run the program using the command mpirun with parameter -n as number of processes:

```
mpirun --machinefile machine.txt -n 10 hello.o
```