Introduction to Cloud Computing

Release Learning

Editor: Gregor von Laszewski

December 03, 2014

CONTENTS

A short video giving a bief overview about this Web site and Cloudmesh is available at  uGIPmiJ0cxg.

CHAPTER

ONE

PREFACE

1.1 Conventions

$ When showing examples of commands, the $ symbol precedes the actual command. So, the other lines are the output obtained after executing the command. An example invoking the ls command follows:

```
$ ls
```

PORTALNAME In some examples we refer to your portal name as the PORTALNAME you have on FutureSystems.

USERNAME In some examples we refer to your local computers name as USERNAME. Your portal name and your local name may be different.

Menu selections: *Start* → *Programs*

Man page: *ls(1)*

1.1.1 Blocks

Hint: This is an important hint.


Hints are represented in simple blocks that are distinguished from the main text

1.2 Using the Notebooks

The material provided in this documentation contains a number of IPython notebooks. Naturally, you do not have to do that, but it may provide you with an easy way to try and run the examples. These notebooks can be executed or accessed through the IPython notebook. We recommend that you set up the cloudmesh server software on the machine where you run the iPython notebooks on. If it is your desktop or Laptop, you can follow the instructions given in the quickstart setup guide.

However, before you can use them you also have to download and install cloudmesh on the machine where you will be running the notebooks from. The installation of cloudmesh is documented elsewhere in this document.

In addition you will need to install the notebook documents locally. You need to run the notebook server from the directory that contains this material. This can be easily checked out from git hub in the following way:

```
$ git clone git@github.com:cloudmesh/introduction_to_cloud_computing.git
```

One you have downloaded the learning documentation form github, you need to cd into the directory with:

```
$ cd introduction_to_cloud_computing
```

Install additional requirements with requirements:

```
$ pip install -r requirements.txt
```

Next you have to compile the information with:

```
$ fab doc.html
```

Now you can start the notebook server with:

```
$ fab doc.notebook
```

You can now visit them and execute them. To view the information locally you can say:

```
$ fab doc.html
```

A link to the list of notebooks will be provided in the sidebar menu.

CHAPTER

TWO

ABOUT

The purpose of this documentation is to provide developers and students with a very simple guide on how to get started to program in the cloud. The documentation is available in a variety of formats including:

- PDF
- epub
- Web pages

The material is based on practical experience that we gained form interacting with undergraduate students starting from the freshman level to graduate students working on a PhD. Naturally, this means that some material may seem to you very simple and at other times we find that some material may be missing or may have changed over time.

It is a pleasure if you can improve this material and potentially contribute your own section to it. You can coordinate your contribution by contacting Gregor von Laszewski (laszewski@gmail.com).

2.1 Independent Study at IU

Gregor is also an Adjunct Associate Professor at Indiana University in the Computer Science Department and thus can supervise students to take independent studies. Independent studies provide an excellent opportunity to engage more intensely in projects related to cloud computing. Independent studies may not just include cloud computing activities but can also be done on other topics related to computer science. You can als propose your own topic and we can discuss if it is suitable. Almost all programming is done in Python, and Javascript. We accept a suitable small number of projects in Java if necessary as part of the topic. The topics are not listed by priority. One or more topic may be covered in the independent study. We will discuss this in a meeting. The amount of work for a 3 hour independent study is the same for a very demanding class in the departments. Expect 15-20 hours of work per week. Those with significant programming experience will have an easier time.

2.1.1 Cloud Topics

- Develop a cloud portal in django replicate significant functionality of our flask portal)
  - must use bootstrap
  - must use jinja2 in addition to djangos rendering engine
  - must interface with our mongodb (not needed for authentication)
  - must provide a user mashup between the user database from django and our own user database (containing contact information)
- Develop a user management system
- Improve the CLoudmesh Portal with Javascript

- Manage 10000 virtual machines
- Develop a PaaS launcher
- Develop an HPC interface
- Develop improvements to the cloud shell

2.1.2 Data Topics

- Develop a program that solves name ambiguity in bibliographic data by investigating a social network graph
- Develop a scalable distributed mongo db for our publication data, conduct performance comparisons for searches

2.2 Student Employment at IU

An independent study is also a precursor to gaining employment as a student with him at Indiana University with us. Students without significant programming experience will not be considered. Preference is given to those that have strong background in python and Javascript. If you do not have such knowledge, we expect that you gain it as part of your independent study. In some cases exceptions are made, this may include students that have programmed significantly in other projects.

Employment is mostly done on an hourly rate. During the time of employment we also recommend that students take an independent study with Gregor. IN some cases this is mandatory to be considered. Thus plan your independent studies carefully.

2.3 Contributing to the Manual

If you have a good chapter that you like to integrate into this manual, please contact Gregor von Laszewski (laszewski@gmail.com).

Suggestions include:

- Development Ecosystem
  - git
  - virtualenv
  - introduction to python
  - introduction to Javascript
  - flask
  - django (we have some material to start with)
  - explain how to create commands in cm with cmd3
- Cloud
  - get an account on FutureGrid (possibly just a link)
  - use nova client from the command shell
  - python example on how to manage vms with cloud mesh API & shell
  - IaaS Intro (pick your IaaS)
  - PaaS Intro (pick your Platform)

CHAPTER

THREE

CONTACT

Gregor von Laszewski laszewski@gmail.com

3.1 Contributors

- Gregor von Laszewski (laszewski@gmail.com)
- Fugang Wang
- Hyungro Lee (hroe.lee@gmail.com)
- Mark Xiao
- Aravindha Varadharaju

CHAPTER

FOUR

RESOURCES FROM THE INTERNET

Throughout the document we will asume certain background knowledge in a programming language. Most of the code is however written in python. For those with solid background in a programming langaugae python can generally be leraned in one or two days. A collection of introductory python (and other programming laguages) are available as part of a

- large collection of free programming books about python

We also found this guid about flask useful that will explain you our GUI framework that we use.

- A flask related book

CHAPTER

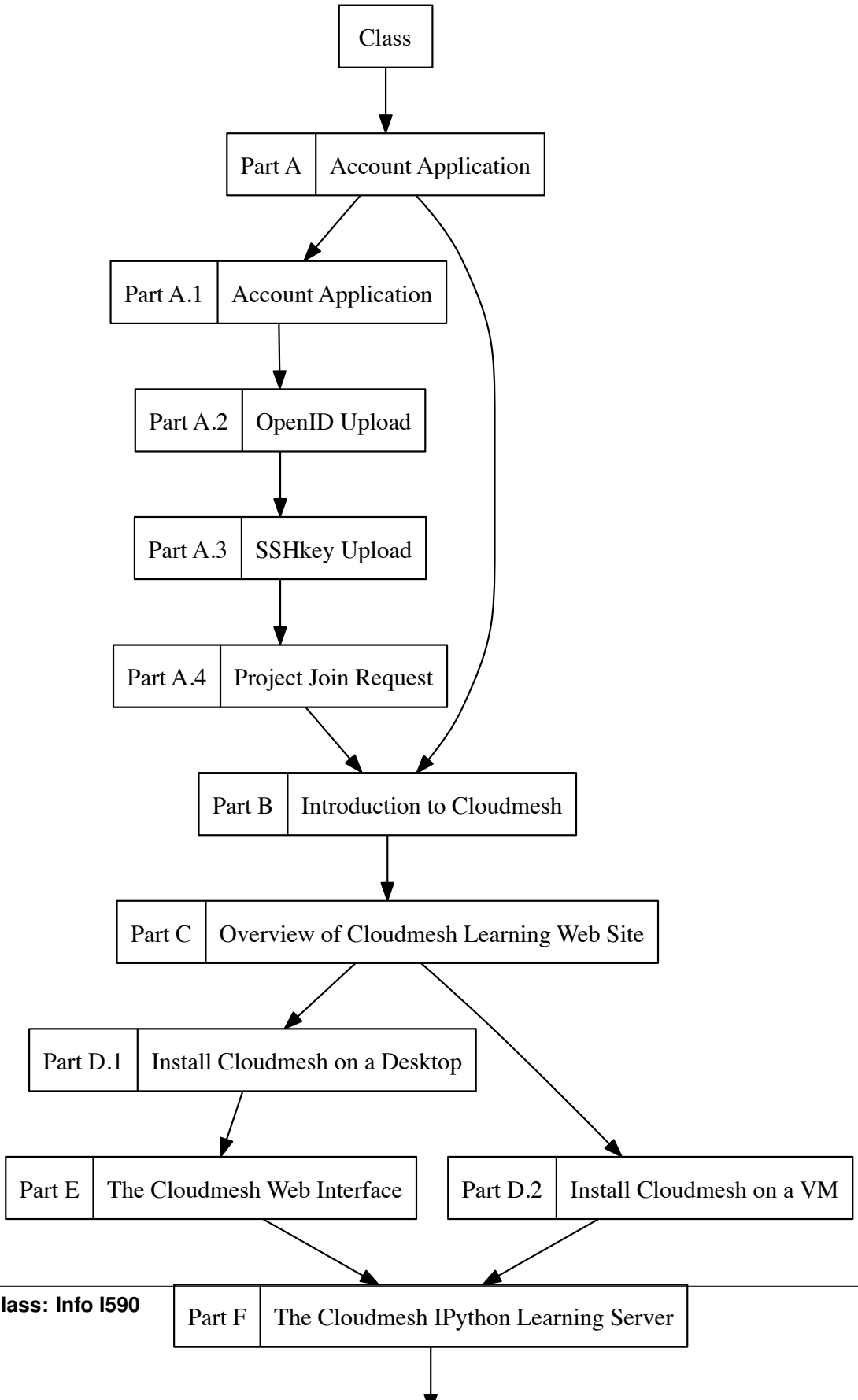
FIVE

CLOUDMESH IN CLASSES

5.1 Class: Info I590

- Fall 2014
- Big Data OpenSource Software and Projects
- http://bigdataopensourceprojects.soic.indiana.edu
- Community Group: https://plus.google.com/communities/117322030211467950209

5.1.1 Video Syllabus



If you attend a class the above video clips are probably sufficient. If you like to know more, please see the videos at *Creating FutureSystems Accounts and Projects*

Exercises

1. Create a portal account on FutureSystems.
2. Identify an interesting project that you can conduct on FutureSystems Cloud, HPC, or Grid resources.
3. Apply for a FutureSystems project or join a project. See section *Join a Project*.
4. Read up on what a ssh key is. See section *s-using-ssh*.
5. Upload a ssh key. Do this via https://portal.futuregrid.org/my/ssh-keys
6. Register your OpenID in the portal (for example if you use google).
7. Upload a 220x220 pixel portrait of yours to the portal. Do this via https://portal.futuregrid.org/my/edit
8. Make sure you have uploaded a small Bio. Do this via https://portal.futuregrid.org/my/Contact

5.1.3 Cloudmesh Introduction

Video	Length	Titles of the Lessons	Description of the Lessons
 njH-HjRMb7V8	28:42 min	A Gentle Introduction to Cloudmesh	This lesson introduces you to cloudmesh. It provides you with an overview, the motivation for cloudmesh and some functionality requirements that motivated the architecture. The lesson also includes pointers to future development of cloudmesh.
 uGIP-miJ0cxg	17:54 min	An Overview about the Cloudmesh Learning Web Pages	This lesson gives a short overview about the web site on which cloudmesh is hosted. It also talks a bit about the architecture.

Excersises

1. What is bare metal provisioning?
2. Is there a difference between the terms raining and provisioning?
3. Identify the different usage of the term provisioning in the community. Give various concrete definitions used in the community. With links.
4. What is a hypervisor?
5. What is IaaS, PaaS, BMaaS?
6. What are prominent IaaS frameworks.
7. What are Software tools you use to conduct Big Data Analysis?
8. If you like to become a contributor to Cloudmesh, contact laszewski@gmail.com.

5.1.4 Cloudmesh Setup

Video	Length	Titles of the Lessons	Description of the Lessons
 lGi-JifD0VgU	17:15 min	Alternative 1: Setup Cloudmesh on a local desktop or laptop.	This lesson explains you how to setup cloudmesh on a local desktop, it will require you to install certain programs on your system. If you do not want to do that, you can use Alternative 2. See Section *Quickstart on your desktop* for more details.
 rcecpgm-47g	32:18 min	Alternative 2: Setup Cloudmesh on a virtual machine.	This lesson explains you how to setup cloudmesh on a virtual machine in the IU cloud. In contrast to Alternative 1 no software needs to be installed on your computer. The video also contains a short introduction to the Web interface. See Section *Quickstart for an Openstack VM* for more details.

Exercises

Choose one of the deployment methods below.

1. Install Cloudmesh on your computer (only if you like to use it on your own machine and are aware that certain programs need to be installed). See Alternative 1.
2. Install Cloudmesh on a virtual machine. See Alternative 2

5.1.5 Cloudmesh Web Interface

Video	Length	Titles of the Lessons	Description of the Lessons
 l_P4G85rysA	15:30 min	The Cloudmesh Web Interface	A lesson on how to use the Cloudmesh Web Interface to manage resources on a Cloud. See Section *Screenshots* for more details.

Exercises

1. Register a cloud (india).
2. Refresh images, flavors and servers
3. Start and delete a vm on india
4. Refresh servers

Note that some features of Cloudmesh have not been activated or may not yet work. Cloudmesh is an evolving project and changes are expected.

5.1.6 Cloudmesh IPython Learning Server

Video	Length	Titles of the Lessons	Description of the Lessons
 1dn_av-zC00	15:30 min	The Cloudmesh IPython Lerning Server	A lesson on how to IPython for directly executing the notebooks contained on on the Cloudmesh learning Web pages.

Exercises

1. Start the IPython server
2. Print the version in IPython (locate the hello notebook)
3. Find additinal notebooks and play with them.

5.1.7 Cloudmesh Command Shell Interface

Video	Length	Titles of the Lessons	Description of the Lessons
 hdq-t-ggkXA	11:58 min	The Cloudmesh Command Shell Interface	This video we will be introducing you to the cloudmesh command shell and demonstrate to you that it is very easy to start virtual machines and access them via a command terminal. The shell provides scripting or the execution of single files. It is fully integrated in a database framework leveraging mongodb.

Exercises

1. Start and delete a virtual machine by using Cloudmesh Command Shell Interface

5.1.8 Cloudmesh Command Shell API

Video	Length	Titles of the Lessons	Description of the Lessons
 mF33LYqC30	9:44 min	The Cloudmesh Command Shell API	In this video we will be introducing you to the cloudmesh command python API and demonstrate to you that it is very easy to start virtual machines and access them via ssh. However python programmer will want to use the Python API. Neverteless this isnterface is nice for quick prototyping.

Exercises

1. Start and delete a virtual machine by using Cloudmesh Command Shell API

5.1.9 Cloudmesh Command Python API

Video	Length	Titles of the Lessons	Description
xOL_-Sfh9MA	14:23 min	The Cloudmesh Command Python API	In this video we will be introducing you to the cloudmesh python API and demonstrate to you that it is very easy to start virtual machines and access them via ssh. We will be using the keys you registered earlier and introduce you to the defaults. This makes it possible to start a VM with only two parameters. We will also teach you how to create public ips and assign them to the vm.

Exercises

1. Start and delete a virtual machine by using Cloudmesh Python API

5.2 Cloudmesh MOOC Shell

cm-mooc provides an easy way to start a Cloudmesh VM on OpenStack India. You can start a virtual machine for Cloudmesh with a single command in *cm-mooc*. You can also enable IPython Notebook on the virtual machine with *cm notebook* commands. You may read the following instructions to enable this program on your terminal.

Tutorial Video Clip: http://youtu.be/kFWGPqHrBCA

5.2.1 Create a FutureSystems Account

First you need to have an account on the FutureSystems portal at

- http://portal.futuregrid.org

We have discussed this in a previous session and you can find out more information in Sthe Section *Account and Project Management*.

Note: FutureSystems Portalname and Project ID For this example we assume you have set the shell variable PORTALNAME to your FutureSystems portal username. This can be done as follwows. Let us assume your portal name is *albert*. Than you can set it with:

```
export PORTALNAME=albert
```

Next you need to login to the india login node with:

```
localhost$ ssh $PORTALNAME@india.futuregrid.org
```

If you need more information about ssh see *s-using-ssh*.

After you log into india you must activate openstack with the following command:

```
india$ module load fg455
```

This setp has to be executed every time yo log into india.

Note: You can also place the lines into your ~/.bash_profile in case you do not want to add the lines by hand every time.

5.2.2 Managing you Key on india

If you have not registered your ssh key to the cloud, you will need to generate one

```
india$ ssh-keygen -t rsa -C $USER-india-key
```

This will generate a key in the default location. You must also specify a passphrase for the key to make the use more secure. Thus you will have a key in *~/.ssh/id_rsa.pub*. This key can now be added to the cloud:

```
india$ nova keypair-add --pub-key ~/.ssh/id_rsa.pub $USER-india-key
```

In case you want to use some more advanced features of cloudmesh you may also want to upload the public key to the FutureSystems portal.

5.2.3 Initializing *cm-mooc*

Note: Do not forget to activate openstack if you have logied in new to india:

```
india$ module load fg455
```

First we initialize *cm-mooc* and open up some ports as part of the openstack default security group. Cloudmesh, IPython Notebook requires to use 5000, 8888 port numbers. We need to add rules for these port numbers.

Note: If you already have *cloudmesh* in your security group, you can skip this step.

```
india$ source ~/.futuregrid/openstack_havana/novarc
india$ nova secgroup-create cloudmesh "cloudmesh ports 5000, 8888"
india$ nova secgroup-add-rule cloudmesh tcp 8888 8888 0.0.0.0/0
india$ nova secgroup-add-rule cloudmesh tcp 5000 5000 0.0.0.0/0
india$ nova secgroup-list-rules cloudmesh
```

Next you simply execute the following commands:

```
india$ cm-mooc start
```

Warning: Please wait approximately 5 minutes after this command. when you log into early the next command will fail.

5.2.4 List the VM Information

You can check the status of the VM by the following command:

```
cm-mooc list
```

The status may report to you active, but that does not mean that all the software is installed yet. So please be patient and wait for some minutes.

5.2.5 Loggin in to the VM

After you have waited for 5 minutes you can execute:

```
india$ cm-mooc login       # SSH to VM
```

This command will start a virtual machine for you that has the software for the class installed. Now that you are logged into the VM you will need to start the ipython notebook server. This is done with the command:

```
vm$ cm notebook create # provide your password to IPython Notebook on the
```

This command will need some input from you and asks you to setup your ipython notebook password as well as information for a self signed certificate

After this step is completed you can exit the virtual machine with the command:

```
vm$ exit
```

Now that you are back on india, you can simply start the notebook with:

```
india$ cm-mooc notebook start
```

This will start the notebook server on your vm while using your password and the certificate you created.

5.2.6 Accessing the notebook

Now you can access the IPython Notebook via a web browser is simple. Just type in the following into your browser url:

```
https://[ip address]:8888
```

If you forgot the ip address you can use the command:

```
india$ cm-mooc info
```

5.2.7 Using the class materials

JavaFiles, please see the tutorial. You can find the class materials in the following directories. Upon your choice of programming language, you can try python or Java examples:

- **IPythonFiles**: directory containing IPython Notebooks for the class fg455
- **JavaFiles**: directory containing cloudmesh Java code
- **cloudmesh**: directory containing cloudmesh IPython Notebooks

The source for these directories is maintained at

- https://github.com/cglmoocs/IPythonFiles
- https://github.com/cglmoocs/JavaFiles
- https://github.com/cloudmesh/introduction_to_cloud_computing

The directory:

```
/home/ubuntu/JavaFiles
```

has the course programs in Java. To view the IPython Notebook programs navigate to the directory with:

```
vm$ cd /home/ubuntu/IPythonFiles
```

Below are the steps to execute the java programs on ~/JavaFiles:

```
vm$ javac -classpath <path to the jar file directory> <ClassName>.java
vm$ java  -classpath <path to the jar file directory> <ClassName>
```

A sample Make file is included in the directory:

```
~/JavaFiles/Section-4_Physics-Units-9-10-11/Unit-11_A-Calculated-Dice-Roll/Makefile
```

For dependencies, please try set your CLASSPATH on ~/Dependencies:

```
vm$ export CLASSPATH=~/Dependencies:$CLASSPATH
```

Similarly for python navigate to home/ubuntu/IPythonFiles directory first cd into the directory:

```
vm$ cd /home/ubuntu/IPythonFiles
```

and than execute the desired program with:

```
vm$ python  <FileName>.py
```

5.2.8 Help

You can see available commands to *cm-mooc* program:

```
india$ cm-mooc -h
```

5.2.9 Deleting the VM

In case you do not need the VM anymore, you can delete the VM with:

```
india$ cm-mooc delete
```

Warning: This is a real delete of your VM with all its contents and data. You want to think twice about if you like to execute the command.

Note: Try Cloudmesh Web Site at http://[ip address]:5000 Your default password is: *cloudmesh* To change the password, try the following commands:

5.2.10 Optional: Starting the cloudmesh server

Warning: If you are not needing the cloudmash server (e.g. you are part of the FG452 project) this part is not needed. You will only use the ipython notebook server

If you laos like to try the cloudmesh server you can srat it on the VM. First make sure you are logged into the vm:

```
india$ cm-mooc login
```

```
vm$ cd ~/cloudmesh
vm$ fab user.mongo # set your password
vm$ fab server.start # restart the Cloudmesh server
```

5.3 JavaFiles

The course provides some examples to run the course Programs in Java. You can run the programs locally or on the cloud with the cm-mooc virtual machine. In this documentation, we focus on how to run the Java programs on the virtual machine.

5.3.1 *cm-mooc* virtual machine on India

If you don't know the running *cm-mooc* on India Futuresystems, please refer cm-mooc. We assume that you have launched the virtual machine and you are staying in $HOME directory on the virtual machine.

Note: To remind you how to use the virtual machine, please see the following commands:

```
localhost$ ssh -X $PORTALNAME@india.futuregrid.org
india$ module load fg455
india$ cm-mooc login
vm$ cd JavaFiles
```

5.3.2 Run Programs

X Window System (X11) should be enabled on your desktop to view plotting data. Java and other libraries are pre-installed on the VM.

Programs Location

On the VM, *JavaFiles* directory contains course programs:

```
vm$ cd ~/JavaFiles
```

Run Each Program

Makefile allows you to compile java programs easily, and to run.

In each directory, please execute:

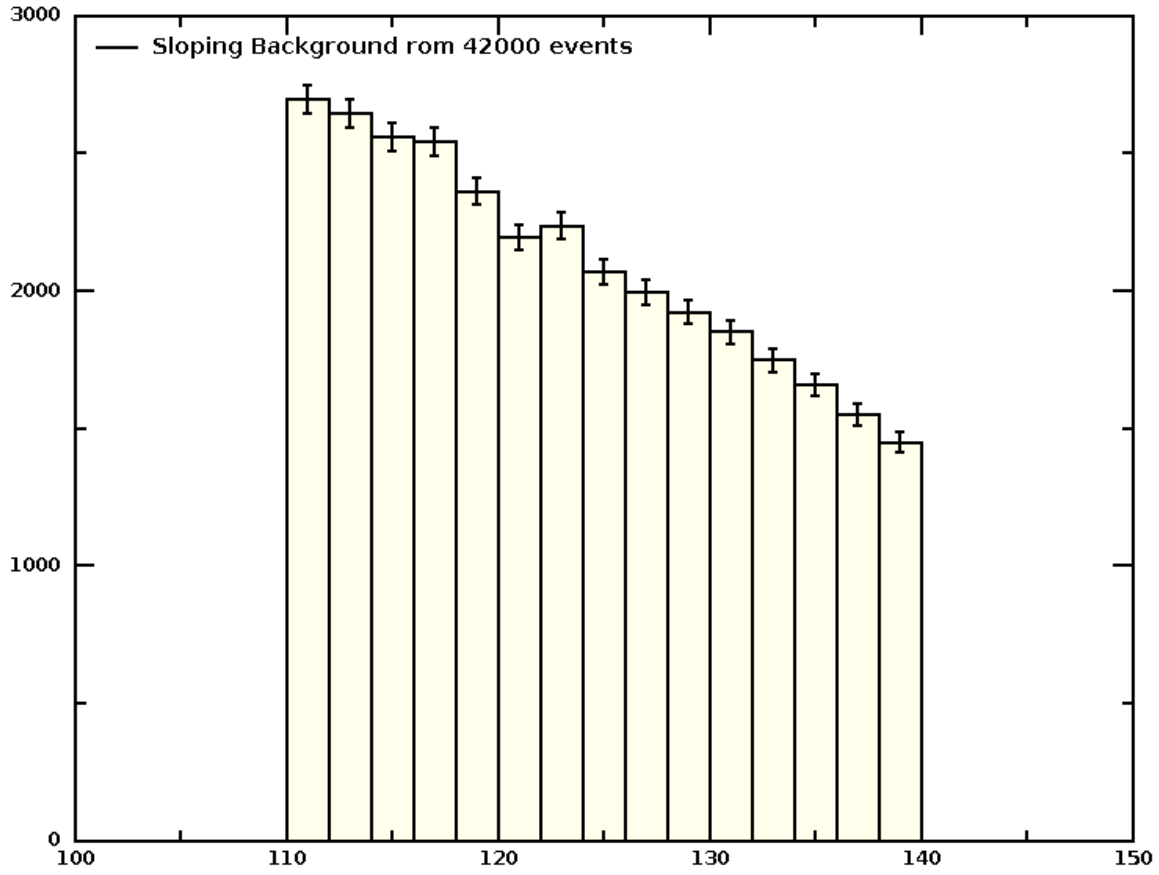
```
vm$ make      # compile
vm$ make run  # run Java Program
```

The Elusive Mr. Higgs

This course program is for the unit 9 in the section 4.:

```
vm$ cd ~/JavaFiles/Section-4_Physica-Units-9-10-11/Unit-9_The-Elusive-Mr.Higgs/
vm$ make
vm$ make run
```

Example of the plotting data is:



Note: Other programs have same templates to compile and run.

5.3.3 Run Programs on a local machine

Download programs and dependencies.:

- Programs:

```
$ git clone https://github.com/cglmoocs/JavaFiles.git
```

- Dependencies:

```
$ git clone https://github.com/cglmoocs/Dependencies.git
```

Note: Make sure that the dependencies are included in CLASSPATH for your Java.

5.3.4 List of Programs

There are 5 programs under ~/JavaFiles directory on the virtual machine.

- In Section 4, Unit 9, The Elusive Mr. Higgs
- In Section 4, Unit 11, A Calculated Dice Roll.

- In Section 7, Unit 19, K'th Nearest Neighbor Algorithms.
- In Unit 27, PageRank.
- In Unit 28, K Means Clustering.

CHAPTER

SIX

CLOUD ACCOUNTS

6.1 Creating FutureSystems Accounts and Projects

Warning: please note that in the videos we refer to FutureGrid, however starting October 1, 2014, the Web site will be called FutureSystems to access resources and apply for accounts. Also be aware that the manual is located at http://mycloudmesh.org/learning

This series of screencasts will walk you through the account and project creation processes of FutureSystems. We have targeted the following areas:

- creating a portal account
- creating a project
- joining a project
- creating a class project
- add a ssh key
- add an OpenID to the portal account

The material will allow you to easily get onto FutureSystems and either create your own project or join an existing one

6.1.1 Videos for users in a class project

Video	Length	Titles of the Lessons	Description of the Lessons
CwH-FaluDgzc	16:33 min	Create a portal account in FutureSystems for class projects.	This lesson explains you how to create a portal account which is the first step in getting access to FutureSystems. You can also view the following videos form FutureGrid as they still apply. For written material, see section *Create a Portal account*.

If you attend a class the above video is probably sufficient. If you like to know more, please see the videos bellow. As you do not have to create a project so you can skip this video.

6.1.2 General videos

Video	Length	Titles of the Lessons	Description of the Lessons
c7mjKI8mJws	5:10 min	Create a portal account	This lesson explains you how to create a portal account which is the first step in getting access to FutureSystems. For written material, see section *Create a Portal account*.
rZzpCY-WDEpI	1:35 min	Upload an OpenID	This lesson explains you how to upload and use an OpenID to login easily into the FutureSystems portal. For written material, see section *Upload an OpenId*.
4wjVwQbOlSU	2:39 min	Upload a SSH key	This lesson explains you how to upload and use a SSH key to login into the FutureSystems resources. For written material, see section *Upload a SSH Public Key*.
DzbLS6iCeTE	6:25 min	Create a project	This lesson explains you how to create a FutureSystems project. For written material, see section *Create a Portal account*.
5xQiPBwt58s	1:28 min	Join a project	This lesson explains you how to join a FutureSystems project. For written material, see section *Join a Project*.

6.1.3 Exercises

Creating FutureSystems Accounts and Projects

1. Create a portal account on FutureSystems.
2. Identify an interesting project that you can conduct on FutureSystemss Cloud, HPC, or Grid resources.
3. Apply for a FutureSystems project or join a project.

   See section *Join a Project*.
4. Read up on what a ssh key is.

   See section *s-using-ssh*.
5. Upload a ssh key.

   Do this via https://portal.futuregrid.org/my/ssh-keys
6. Register your OpenID in the portal (for example if you use google).
7. Upload a 220x220 pixel portrait of yours to the portal.

   Do this via https://portal.futuregrid.org/my/edit
8. Make sure you have uploaded a small Bio written in third person.

   Do this via https://portal.futuregrid.org/my/Contact

6.2 Account and Project Management

Page Contents

It is very easy to obtain a project and account on FutureSystems. However, you need to get in contact first with FutureSystems to identify if you are allowed to use this research environment.

While it is possible to just execute the three steps in our quickstart guide, we have provided a more in-depth description based on user feedback. You have certain responsibilities including **managing project memberships** and **reporting results** that must be conducted while you use FutureSystems, thus it is a good idea to read this section carefully. At the end of the section we also provide a mini-FAQ of information that may be of help with respest to issues you may have overlooked or that have not yet been answered. Some screencasts about this topic are available in section *Creating FutureSystems Accounts and Projects*.

6.2.1 Terminology

Portal Account: A portal account is necessary to communicate information about yourself to the FutureSystems team so that they can verify your identity and that you are a community member who has a need to use FutureSystems. Once you have a portal account approved you can apply for a project.

Resource Account: A portal account will not give you access to the FutureSystems resources. You will have to go to the portal and join a project, or create a new one.

Valid Project: At least one project you belong to must be valid. A valid project is one that is approved by a committee. If you are not in a valid, active project your access to FutureSystems will be blocked. Projects that do not report any progress will be blocked after a while. Note that you agree to update regular results via the portal as part of your agreement to use FutureSystems.

Project Lead: The Project Lead submits a project and is responsible for updating the project page and project members. There is exactly one Project Lead.

Project Manager: The Project Lead can assign editing and management roles to a Project Manager. He will have the same rights in regards to updating page content and members as the Project Lead.

Project Member: The Project Lead or Manager can add additional members to the project. A member must have a portal account.

Project Alumni: Sometimes project members leave a project before it is completed. Such members are no longer part of the project and should not have access to the project. However they can still be acknowledged to have played a role as part of the project by being placed as a member of the alumni role.

Uploaded SSH Key: As you are using remote compute resources, we will require that you are very familiar with ssh key management. You will need to upload a public ssh key to the portal so that we can create an account for you and use that public ssh key to allow you to log in to use the resources.

6.2.2 Quickstart

If you have never created a project on FutureSystems, we recommend that you **do not use the quickstart guide** and instead read the full documentation. To remind you what you have to do here are a couple steps that you need to do:

1. Create a Portal account and wait for confirmation via e-mail
2. Join a project or create a new project and wait for confirmation. Note that you must com municate with the project lead first before you can join a project.

6.2.3 Project Management

Create a Portal account

In order to utilize **any** FutureSystems resource, you must possess a FutureSystems **portal account**. Thus, *apply for your portal account* before you attempt anything else. This account is used to gather some information that we will use in the next steps. You must make sure that the information is complete before you proceed to the second step. FutureSystems performs basic verification of the information you provide when creating an account, so it may take a little while before your account is approved. Once you have a portal account, please proceed.

Please note that you cannot access FutureSystems resources until you complete the next steps.

Hint

It may take a day or two to get a portal account. Portal accounts will not be created over the weekend.

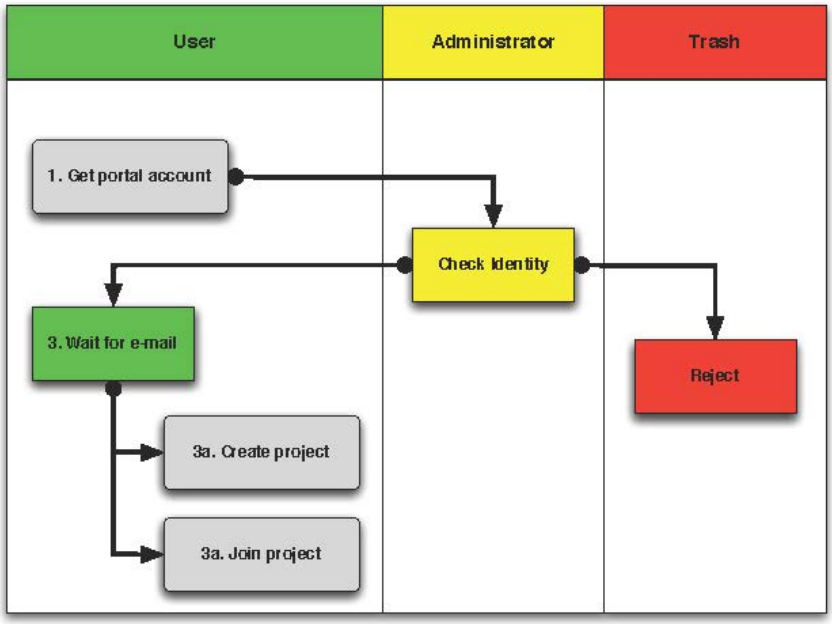
Here are a few tips that make it easy for you

- **On the portal's main page at https://portal.futuregrid.org appears a** *Portal* → *Register* link.
- **Following you will be able to Create a new account on the** portal.
- Fill in **ALL** fields as much as you can.
- Note that fields with * are mandatory
- **It is important that you specify your address information** completely.
- **If you are a graduate or undergraduate student please fill out your** advisor's contact information in the field specially dedicated for it. If he has a FutureSystems Portal name, please include his portal name if you know it.
- **If you have an e-mail address from your institution, we ask that you** use this address instead of one from gmail, hotmail, or other e-mail services that we cannot trace back to your name or institution.
- **Usage of all non institutional addresses will prolong the** application process.
- **Please note that creating a portal account does not give you access** to any FutureSystems resources.

- **Please remember that checking your information will take time. Thus** we recommend that you wait till you get a message that tells you that your portal account has been approved. Then continue to The next step. We are not conducting any portal approval outside of 10am-4pm EST. If there are no problems verifying your information your approval will take 1-2 days; if we have problems verifying your data or something else is not right your approval will be delayed. If you appear to be a spammer we will not notify you.
- If you are teaching a class, we have some special instructions for you in Section *s-account-class*.
- After your account has been approved, you can correct the information as part of the portal account User Profile Management.

Table 6.1: Legend

State	Description
1. Get Portal Account	Apply for a portal account at https://portal.futuregrid.org/user/register
Check Identity	Administrator checks the data submitted.
Reject	Rejected accounts will be deleted without notification.
2. Wait for e-mail	Wait for the e-mail that approves your portal account. If you have not heard from us within 2 buisiness days use the help form on the portal to contact us.
3a. Create Project	Create a new Project.
3b. Join Project	Join an existing Project.



Create a Project

To apply for a new project, fill out the project creation form. Through this form we gather some important information about the project so that we can review it for approval. This information is used to report and document to us as well as to our sponsors, to state which activities are conducted on FutureSystems. The more precise you are in your descriptions and filling out the forms the better we can highlight your project. Once a project is approved, project members can join a project. This must be conducted by the project lead.

It is mandatory for the project lead to agree to certain reporting requirements so as to provide information to FutureSystems. He will be responsible to make sure that they are completed and also implemented with the users joining the project. Thus the user is responsible to comply with the terms of the project in regards to reporting and acknowledgments in case of publications. Each project Project Lead has the responsibility to communicate such requirements to the members and managers. The project agreements override the individuals agreement.

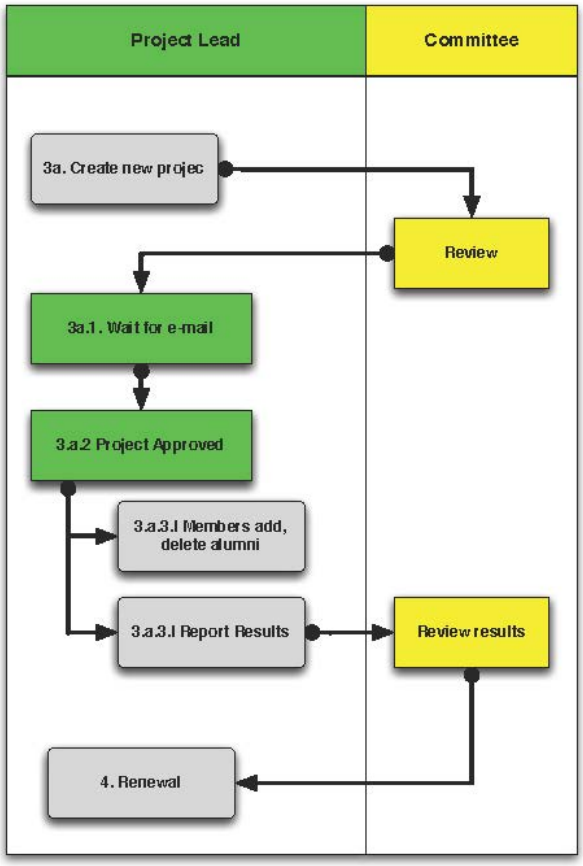


Table 6.2: Legend

State	Description
3a. Create new Project	Fill out the project form at https://portal.futuregrid.org/node/add/fg-projects
Review	Committee reviews the project and corresponds with project lead to improve
3a.1. Wait for e-mail	Wait for an e-mail that you have an account
3a.2. Project Approved	The project has been approved.
3a.3.i. Members add, del., alumni	Manage the project members
3a.3.ii. Report Results	Make project member Alumni
Review Results	Committee receives results for review
Renewal	Renewal of the project

Example Project

When applying for a project, you may directly visit the project creation page and fill it out. However, you may find it useful to prepare a separate (ASCII, or MS word) document and take advantage of spelling and grammar checkers. Furthermore, as filling out the form may take some time, it seems best to just copy and paste from your document into the form. This way you may avoid issues such as power failures or network interruptions which could cause you to

lose the information that has been entered on the form but not yet submitted.

Examples for a project can be found also on the portal itself when looking at the list of projects conducted on Futuregrid:

- All projects: https://portal.futuregrid.org/projects/all
- Class Projects: https://portal.futuregrid.org/projects/keywords/course

Recently we have increased the requirements for project approval. Hence it is important that you write a couple of paragraphs in the application. A single sentence such as *I want to learn cloud computing* will typically no longer be approved. Please examine a sample class project which could be an inspiration for your own class projects (`project.txt`):

```
Title: Course: Example Course On Advanced Cloud Computing

Project Keywords:
   Course, Cloud, OpenStack, Eucalyptus

Project Lead:
   Gregor von Laszewski (portalname)

Project Manager:
   Gregor von Laszewski (portalname)

Project Members:
  Fugang Wang (use portalname)
  Albert Elfstein (use portalname)

Project Alumni:

Project Orientation: *
- [ ] Research
- [x] Education
- [ ] Industry
- [ ] Government

Primary Discipline: *
  Computer Science

Abstract: *
=========

Note: this is an example project and is not a real project,
although the contents presented in this material is available.

This course will introduce the students at Indiana University as
part of the Summer Semester 2012 into the essentials of Cloud
Computing and HPC. We will start the course by teaching the
students python within one week. As cloud computing framework we
have chosen OpenStack, as it has become one of the ubiquitous IaaS
frameworks and is available on FutureSystems. Additionally, we will
teach the students how to program a simple MPI application so that
they can further develop the virtual cluster code available from
github (https://github.com/futuregrid/virtual-cluster).  We will
compare the performance between the virtualized and non virtualized
environment as develop with the help of our cloud metrics system a
scheduler that enables us to use bare metal provisioned clusters
and virtualized clusters on-demand based on resource requirements
and specifications. We are aware that the FutureSystems team is
```

```
developing such an environment, and would like to join the efforts
throughout our course with the contributions conducted by the
students.

Course Dates:

This class will be taught in 10 weeks as part of the Indiana
University CS curriculum. The following dates are important

      Start: July 13, 2013
      End:   Sept 23, 2013
      Extension: 1 month for students with programming in-completes.

Course Outline (tentative):

   1. Introduction and Overview
   2. Essential Python for the Cloud
   3. Introduction to OpenStack
   4. Programming OpenStack
   5. Programming a HPC Cluster
   6. Creating a Virtual Cluster
   7. Performance Comparison
   8. Cloud Metrics
   9. Cloudmesh
   10. Joining FutureSystems Software Developments

Grading Policies:
   Class participation and contribution: 5%
   Homework assignments, reading summary, and paper presentation: 50%
      Programming assignments: (30%)
      Reading Summaries: (10%)
      Paper Presentation: (10%)
   Course Project: 50%
      Proposal: (10%)
      Midterm Presentation: (10%)
      Final Presentation and Demo: (15%)
      Final Report: (15%)

   Note:
     Homework and programming assignments are due by 11:59pm
     Thursdays (unless announced in class otherwise). Late homework
     (non-programming) will NOT be accepted. Late program penalty
     is 10% per day, according to the timestamp of your online
     submission. Only when verifiable extenuating circumstances can
     be demonstrated will extended assignment due dates be
     considered. Verifiable extenuating circumstances must be
     reasons beyond control of the students, such as illness or
     accidental injury. Poor performance in class is not an
     extenuating circumstance. Inform your instructor of the
     verifiable extenuating circumstances in advance or as soon as
     possible. In such situations, the date and nature of the
     extended due dates for the assignments will be decided by the
     instructor.

 Please note that FutureSystems does not approve accounts on the
 weekends. Regular support hours are Mo-Fri 9am - 5pm. Please
 note that answering support questions does take time. Do not
 start the night before the homework is due. Plan your
```

```
 programming assignments to be done early.

Intellectual Merit: *
====================

The course will be introducing the students to cloud computing and
will also be used to derive new class materiel that we will be
using in subsequent lessons.

Broader Impact: *
===============

This class will be educating a number of students in cloud
computing programming. Cloud computing is an important factor in
job availability after graduation of students, thus this course
will be useful to increase marketability of the students. In
addition we have in the past also been able to increase
participation of minority students. In the past we had 10 minority
students and 9 female students taking this class. We intend to work
together with Gregor von Laszewski and improve the FutureSystems
manual and to make our course material available via FutureSystems
through its github and community portal pages.

Scale of use: *
=============

We anticipate the course will have 30-35 students. The course
will be using OpenStack and HPC compute resources and requires for
selected students access to bare metal provisioning.  The course
will not require to run computationally intense
applications. However, we require that students be able to run up
to 30 VMs at a time. We know that this may in peak hours be beyond
the capabilities of FutureSystems and are advising our students to
kill machines if they are not used. The maximum duration of a
single VM will typically be less than 5 minutes.

Results:
========
```

Join a Project

To join an existing project, ask the project lead or project manager for that project to add you to their project using your portal account name. However for most projects there is an easier way if the project is set to "accept public join request", you may also send a request in the portal. To do this, first view the project list and go to the project detail page by clicking the project title. If the project is set by the project lead to "accept join request", then you'll see a large gray 'Join this project' button in the upper right corner of the page. Click the button to send the join request to the project lead and manager so that they can process your request:

Project Number:
FG-

Project Lead:

Project Manager:

Institution:

Project Status:
Active

- Join this project
- Manage Join Requests
- View Project Members
- View Project Details
- Edit Project Details

The entire process looks as follows:

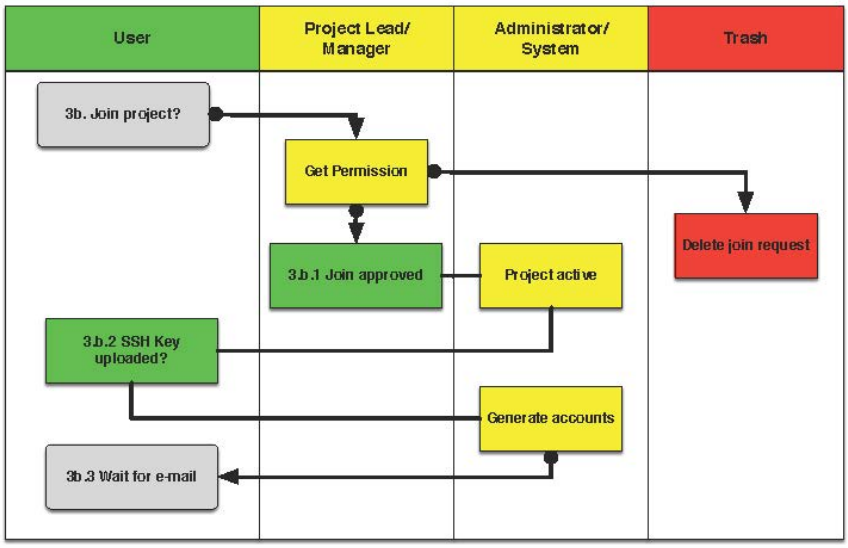


Table 6.3: Legend

State	Description
3b. Join Project?	Join an existing Project.
Get Permission	Get permission from the project lead or manager to join the project.
Delete Join Request	Project leads will carefully evaluate if the person requesting to join is eligible. If not join requests will be deleted without notification requires to those that want to join.
3b.1. Join Approved	The project lead has approved that you join the project. Go to the project page and press the join button/link.
Project Active	Checks if the project is active the project.
3b.2. SSH key uploaded?	You must have uploaded your ssh key to use FutureSystems resources
Generate Accounts	Generate accounts to resources
3b.3. Wait for e-mail	Wait for an e-mail that you have an account

Delete or Deactivate a user from a Project

Deletion of a user can be a complex process if a user has used FG resources. However, the following is for most project leads sufficient: To remove a user from your project you simply can edit your project page and remove the user name from the list of members or alumni. If the user is in no other valid project she will no longer be able to use FG. If the user really needs to be removed from the system or from the portal, please fill out the help form with the username and the reason why he should be removed. Naturally, if you detect that a user is acting maliciously, please inform us immediately. We will disable access. Put in your subject line the prefix URGENT.

Reporting Results

It is important to regularly report results of your projects to FutureSystems. Please fill out this section and report upon the achievements of this project. You find your projects in a

- list maintained in the portal

Also upload references that you have produced for this project. You can upload when visiting your project and using the plus button in your projects. The reference will then be added and added for you to the portal.

Close a Project

Closing a project is an important responsibility of every project lead. Incase you forget which project you want to close, you can find the

- list of projects you lead or manage

on the Portal. Once you visit one of them, you have the option to add results. Scroll down until you find the section "Project Results". Please fill out this section and report upon the achievements of this project. Please also upload references that you have produced for this project. In each case, please make sure that you only upload/report references directly related to this project. After you have requested a project closing, our project committee will work with you to make sure we have sufficient results from you. Once a project is closed, all its members will be notified. The committee might ask you for additional results even after the project is closed.

6.2.4 Upload a SSH Public Key

In order to be able to log into the started VMs, among other purposes, you need to provide FG with a secure-shell (ssh) public key. If you are already a frequent user of ssh, and have a private and public key pair, it is perfectly reasonable to provide your public key. It's *public*, after all.

To upload the chosen public key:

1. Copy your public identity into your system clipboard.
2. Visit the ssh-key panel of your account.
3. Click the link that says Add a public key.

If you are not familiar with ssh keys we have provided a more elaborate section about *s-using-ssh*

Changes to keys will take up to 1 hour to propagate through the system services. You are not allowed to use password less keys. Your account may be deactivated if you violate this policy.

6.2.5 Upload an OpenId

Often users may not remember the password or username of the FG portal. However, they may have an easier time to remember their openid from for example google. It is possible to use your openid account and register it once you gain access to the portal. Please visit your

- OpenID Page

to add your favorite OpenID. For example, to add your Google OpenId you simply click on the Google icon.

6.2.6 Accessing FutureSystems Resources

To access and use resources, you must

- have a portal account
- be part of a valid project
- have uploaded a public key to the portal that you will use to log into some of its resources.

Once these conditions are met, you will be able to access the resources and services that your project has requested and been authorized to use. See the section *s-services* for a list of FutureSystems resources and services. This includes cloud and HPC resources. Accounts to these resources will be automatically generated once you have conducted the above steps. The turnaround time for you getting access to the system is typically between 30 minutes and one day.

6.2.7 Manage a Class on FutureSystems

If you teach a class using FutureSystems resources we recommend you do the following:

1. Create a portal account if you do not have one.
2. Apply for an educational project, carefully filling out the form including how many students, broader impact, such as support of minorities, what will be learned, the course syllabus if available, a link to the course web page if available, the duration of the course including a time when the course is completed. We typically add a month so that incomplete projects can be completed easily.
3. Make sure you enable the join button of the project, this will allow your students to join via a button click and you can easily approve or reject join requests. Come up with a "signup code" to be shared with the students.

4. Give your students the signup code that you have chosen in the previous step. Communicate the signup code to the FutureSystems support team via a ticket submitted through https://portal.futuregrid.org/help. Make sure you specify your project number. Often it is also helpful to send a list of students to us so that, it is easier for us to assist them during the application process.
5. Make sure your students sign up for a FutureSystems portal account and that they specify their profile information precisely. This information is used later on to grant students accounts on FutureSystems. Accounts will not automatically be created just because a user has a portal account. Have the students add you as their advisor in the advisor textarea.
6. Remind the students to add their public ssh-keys to the portal. Some students may not have the knowledge what this means or what this is good for. We recommend that you in the first class teach the why they need to do this and how the can do this. In the portal users can add ssh keys when they go to the my portal section.
7. It will take some time for the accounts to be created after an ssh key is uploaded, and the student is added to your approved projects. Communicate to the students to wait. We only approve accounts during business hours and it can take up to 24 business hours. Business hours are Mo-Fri 9am - 5pm EST. We will not answer any questions on the weekends.
8. Once a student has an account on the portal, please make sure you add the student to your project. This is important as only people that are assigned to a valid FutureSystems project can have accounts on FutureSystems resources. Your project will by default have a project join request, which makes it easier for students to join your project. Provide the link to the students so that they can join. A convenient management button is provided where you can verify that the student is indeed a person that is to be part of your project. The join button can be disabled by you and you could instead add your students while entering their portal names.

   Hence, you will be able to manage the joining of students yourself. Be careful that you only join those students that are in your class. Please remember that a signup code is not really secret and that students may exchange the code with others. Thus it is a good idea to still verify if the user with the signup code is a member of your class. Also be reminded that some students forget to specify the signup code at time of their account creation. You have to deal with such forgetful students as a signup code cannot be added.
9. If the student roster is changing, just edit the project details and add/remove them or move them to the alumni status.
10. If student projects are due on Mondays remind them not to start their project on Sunday night incase they find out they do not have an account. Generally we recommend to make due dates of projects to be Thursdays till 5pm or Friday mornings. Be reminded that on the first Tuesday of each month all machines will be shut down and all unsaved running VMs or ongoing work may be lost. Please plan around this.
11. We have created some forums for the three services that you can find at https://portal.futuregrid.org/forum. These forums are read by the experts and the staff. We can create a forum for your class if you like directly on the FutureSystems portal.
12. In case you need more direct support, do not hesitate to ask for help https://portal.futuregrid.org/help
13. Make sure you write a results section after your class is over.

6.2.8 Mini FAQ

- Which Projects Do I Participate In?
- How can I *Join a Project*?
- How can I add people to a project?
  - Go to your project, select the add member link and add the user portal names. Alternative have your users use the join button and you use the manage button.
- Why Do I See in the Project Table "Please Sign Up"?

  - If you are the owner of a project and see this information under project lead or manager, you may not yet have signed up for a portal account. Please sign up for one, and we will change it in the project view for you.

- Why do I need to provide the email address from my university?
  - It may take longer to approve your account. See *Create a Portal account*
- How long will it take for my portal account to be approved?
  - If you did everything and we can verify you exist two business days.
- How Do I Get an Account for OpenStack? see *Accessing FutureSystems Resources*
- How can I *Delete or Deactivate a user from a Project*
- How do I *Upload a SSH Public Key*
- How do I get a user account on FutureSystems resources? see *Quickstart*

CHAPTER

SEVEN

CLOUDMESH

7.1 Cloudmesh Overview

We have provided some documentation about cloudmesh at http://cloudmesh.github.io/index.html.

Cloudmesh is an important component to deliver a software-defined system – encompassing virtualized and bare-metal infrastructure, networks, application, systems and platform software – with a unifying goal of providing Cloud Testbeds as a Service (CTaaS). Cloudmesh federates a number of resources from academia and industry. This includes existing FutureSystems, Amazon Web Services, Azure, HP Cloud, Karlsruhe using various technologies.

An high level architectural image is provided at

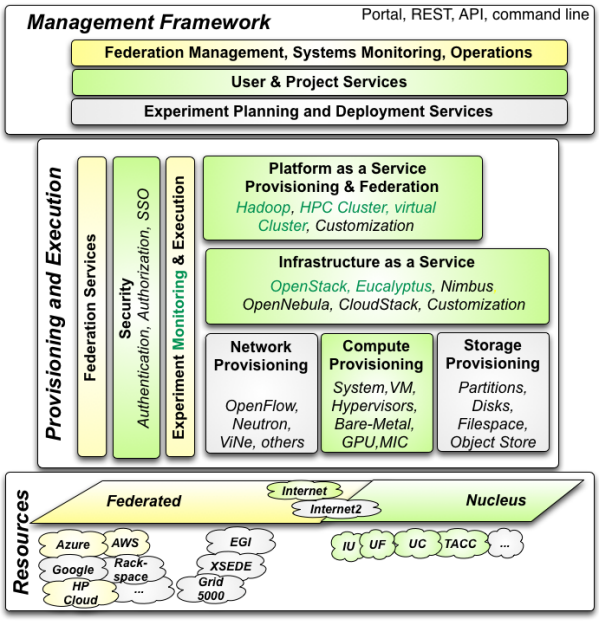


Figure 7.1: **Figure:** *High level architecture of Cloudmesh*

The three layers of the Cloudmesh architecture include a Cloudmesh Management Framework for monitoring and operations, user and project management, experiment planning and deployment of services needed by an experiment, provisioning and execution environments to be deployed on resources to (or interfaced with) enable experiment management, and resources.

Before continuing we recommend that you look at the following sections on the cloudmesh web page:

- http://cloudmesh.github.io/cloudmesh.html

- http://cloudmesh.github.io/cloud.html
- http://cloudmesh.github.io/rain.html
- http://cloudmesh.github.io/hpc.html

In this document we like to focus on the actual implementation of cloudmesh and how the various components are structured. For this purpose we have redrawn the Figure we pointed out earlier while focussing on a number of subcomponents that we will be looking more closely into.

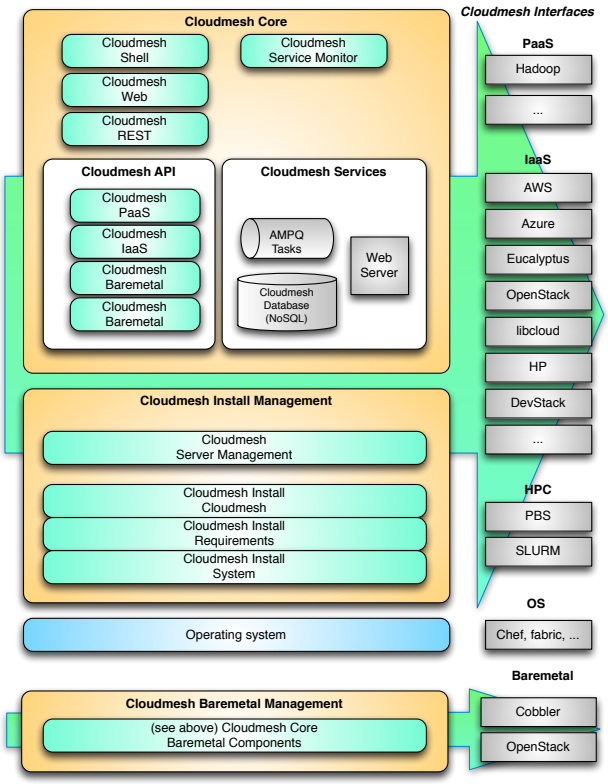


Figure 7.2: **Figure:** *High level architecture of Cloudmesh*

Typo: there are two baremetal boxes in cloudmesh cor. one should be HPC

This diagram highlights some important information which we describe next

7.1.1 Cloudmesh Main Components

The main cloudmesh component include:

- Cloudmesh Install Management: which allows to easily install cloudmesh on a given operating system. This can also include a virtual machine.
- Cloudmesh Baremetal Management (currently part of core): which allows the deployment of an os via bare metal through cloudmesh. The important differentiation to other systems is that users can be authorized to conduct bare metal provision based on service policy descriptions. The user may not be the administrator of the machine.
- Cloudmesh Core: which contains the major components to interface with external subsystems to conduct bare matel provisioning, interact with IaaS such as virtual machines, HPC queues, or the bare metal provisioning services.

7.1.2 Cloudmesh Install Management

- Conducted in three phases
- **Phase 1:** prepare the system. Te OS may have missing packages. the program will install all missing packages for cloudmesh. This step requires typically sudo permissions.
- **Phase 2:** Install the python requirements. As cloudmesh is mostly developed in python, this step installs all necessary python libraries.
- **Phase 3:** Install the cloudmesh programs. This Step installs the cloudmesh program in the python library. (We use virtual env for our development)

7.1.3 Cloudmesh Service Management

After all the software is installed, we can start up the various cloudmesh services. This includes

- **Cloudmesh Database**: A NOSQL database in which we record which virtual machines run on which IaaS. This allows us to have a federated view of the heterogeneous clouds.
- **Cloudmesh Web Service**: Provides a Graphical user interface to manage virtual machines and HPC tasks
- **Cloudmesh Task Service**: As cloudmesh is a multi user systems many tasks need to be handles in parallel. To achieve this we are using an AMPQ queue and coordinate the execution of managing multiple virtual machines for multiple users.

7.1.4 Cloudmesh Use Mode

Base on the rich service model in Cloudmesh we are able to start cloudmesh either in a

- standalone mode or in a
- hosted mode for multiple users.

For development purposes most users will want to run the cloudmesh services in standalone mode. This enables you to test out cloudmesh without interfering with other users. It also allows you to be completely responsible for your credentials without relaying them through a third party.

7.1.5 Cloudmesh Baremetal

Cloudmesh contains an interface to cobbler for its bare metal services. However the important feature is that it also contains a very small abstraction interface to bare metal provisioning. This will allow us to integrate with other bare metal provisioners and enable for example the use of OpenStack Ironic once it is deployed for example on FutureSystems.

7.1.6 Cloudmesh HPC

Cloudmesh provides an easy tou use API and GUI to HPC queues. It allows simple display of queues. As FutureSystems shares on some systems the queue manager it also seperates the queues appropriately and displays them accordingly. The API returns the job and queue information as python dicts.

7.1.7 Cloudmesh IaaS

Cloudmesh contains an abstration to interface with arbitrary IaaS farmeworks this includes

- Azure
- AWS
- OpenStack
- Eucalyptus
- clouds that can be accessed through libcloud

The important differentiation to other frameworks is that it is not just capable of interfacing with libcloud to a remote cloud but it is possible to provide interfaces while using the native protocol. This has greatly helped in debugging real clouds as for example some features are not properly exposed through libcloud or EC2 compatible mechanisms. It also protected us from several changes that took place during the various versions of OpenStack. Our OpenSTack library interfaces directly with the OpenStack REST services

7.1.8 Cloudmesh Web

While other clouds focus on their own infrastructure, Cloudmesh provides a user interface with federation capabilities to display and interact with heterogeneous clouds. In addition information between these clouds is not hidden behind a compatibility library such as libcloud or a cloud standard, but uses instead the natively available information. This allows developers to interact and inspect information on a different level than just being able to start and stop virtual machines. Interfaces to HPC queues are also available. The Web services interfaces with the Task and Database Services.

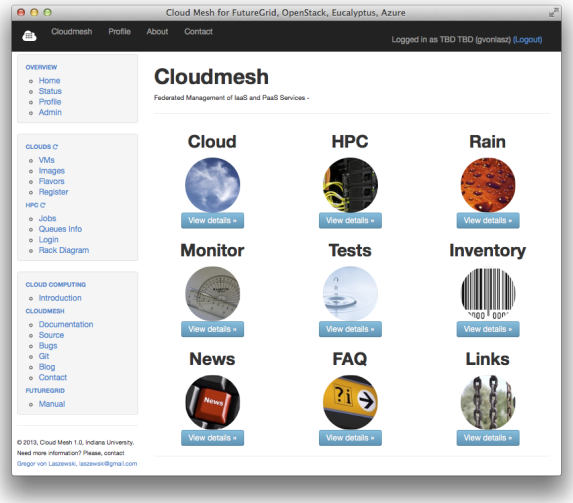


Figure 7.3: **Figure:** *The Cloudmesh Web Interface*

7.1.9 Cloudmesh Shell

For experiment management it os often not sufficient to just provide a GUI interface but to be able to script how virtual machines are coordinated. This can be done with our cloudmesh shell that similar to matlab has its own shell

environment, but can also be simply be called as a command on a regular Linux terminal. The Cloudmesh Shell services interfaces with the Task and Database Services.

```
cm cloud list
```

```
+-------------------+--------+
| cloud             | active |
+-------------------+--------+
| alamo             | True   |
| aws               |        |
| azure             |        |
| dreamhost         |        |
| hp                |        |
| hp_east           |        |
| india             | True   |
| india_eucalyptus  |        |
| sierra            | True   |
| sierra_eucalyptus |        |
+-------------------+--------+
```

7.1.10 Cloudmesh API

A convenient API is presented to interface to cloudmesh in python (shown here how to select an image on the cloud india):

```
#
# IMPORTS
#
import cloudmesh

#
# INITIALIZATION
#
mesh = cloudmesh.mesh("mongo")
username = cloudmesh.load().username()
mesh.activate(username)

#
# GETTING INFORMATION ABOUT THE IMAGE
#
image=mesh.image('india','futuregrid/ubuntu-14.04')

print image
```

7.1.11 Cloudmesh State

As the Shell and the Web Service interact with the Database and the Task Services, the status after a refresh is synchronized between them. This means if I start a virtual machine with the command shell I can see it in the Web after a refresh and vice versa. Default values are shared and the interaction between Web and Shell is seamless.

The emphasize is here on managing multiple machines and the start of a VM can be done with a single click in the Web or with a single command without parameters in the shell. This is in contrast to other frameworks that do not make use of extensive default management for repetitive interactive experiments.

7.1.12 Tutorials

We have provided a number of tutorials through IPython notebooks that you can follow. A setup guide is available that documents the installation of cloudmesh through a single curl call. The tutorials will show you each of the three different interfaces including:

- the Python API
- the Web GUI via the Web browser
- the command shell

The examples focus on displaying information and managing virtual machines.

A list of all notebooks is available. The list will be expanded, and we would be happy if you contribute to them with your own suggestions. If you think we need to show a particular feature, please let us know. We wil try to add it.

7.1.13 Development and Transition to FutureSystems

- Due to the transition of FutureGrid to FutureSystems, we have limited our tutorial activities in regards to baremetal provisioning
- We are focussing our current efforts on the development of our PaaS launcher that interfaces with chef and ssh to deploy platforms on other resources.

7.2 Cloudmesh Setup

Cloudmesh can be setup to on your local environment in single user mode, or as a hosted service for multi-tenancy use. Based on your needs you may decide which way to set it up.

For development and class users we recommend the local setup.

7.2.1 Quickstart on your desktop

A video bout the quickstart instalation is available on  lGiJifD0VgU

Warning: this tutorial is for the new FutureSystems infrastructure. However at this time we still use FutureGrid. Please replace all occurrences of FutureSystems with FutureGrid.

This quickstart is designed for Ubuntu 14.04 and OSX.

Note:

FutureSystems Portalname and Project ID For this example we assume you have set the shell variable PORTALNAME to your FutureSystems portal username. This can be done as follwows. Let us assume your portal name is *albert*. Than you can set it with:

```
export PORTALNAME=albert
```

We also assume that you have a project id that you set to:

```
export PROJECTID=fg101
```

if it is the number 101.

We recommend that you use virtualenv to provide an isolated environment for cloudmesh. We assume you create one called ENV and activate it:

```
$ virtualenv ~/ENV
$ source ~/ENV/bin/activate
```

First you need to download the code from github. We assume you have git installed:

```
git clone https://github.com/cloudmesh/cloudmesh.git
```

Next, you need to install a number of required packages with the following commands:

```
$ cd cloudmesh
$ sudo ./install system
$ ./install requirements
```

Note: on OSX you can omit the sudo.

To get access to IaaS cloud platforms, you need to create locally a new user that has access to various clouds. This can be done with:

```
$ ./install new
```

The next steps will deploy the cloudmesh code into the virtualenv library path:

```
$ ./install cloudmesh
```

Note: This step is optional but highly recommended for users.

In case you have accounts on the IU machines you can also obtain pre-configured cloud rc files from them. To test if you have an account and have set it up correctly, please login to the machine india:

```
$ ssh $PORTALNAME@india.futuresystems.org
```

If this does not work, you may not have uploaded your public key to FutureSystems portal at

- https://portal.futuresystems.org/my/ssh-keys

Once this step is completed, you can create the configuration files as follows:

```
$ cm-iu user fetch
$ cm-iu user create
```

At this time we like you to edit some information about yourself in the cloudmesh.yaml file. Choose your favorite editor:

```
$ emacs ~/.cloudmesh/cloudmesh.yaml
```

Change the values TBD that you find here with values that describe you.

As you will need at one point to login into virtual machines you will need a key that cloudmesh can use do to so. We assume you have a public key generated in your .ssh directory in the file:

```
$ ~/.ssh/id_rsa.pub
```

If you do not have such a key, you can generate it with:

```
$ ssh-keygen -t rsa -C $PORTALNAME-key
```

The next steps will deploy the cloudmesh database:

```
$ fab mongo.reset
```

We add the key to the database with:

```
$ cm "key add --keyname=$PORTALNAME-key ~/.ssh/id_rsa.pub"
```

where PORTALNAME is your name for the FutuerSystems portal.

You may next need to specify your default project if you have not yet done so:

```
$ cm project default $PROJECTID
```

where PROJECTID is your default project id from FutureSystems e.g. fg455 as an example.

To start Cloudmesh use:

```
$ fab server.start
```

Now you can test the service by visiting the web interface at http://127.0.0.1:5000. We have a convenient shortcut for this by typing:

```
$ fab server.view
```

Alternatively you can use the cloudmesh shell by invoking the cm command via a terminal:

```
$ cm

=======================================================
/ ___| | ___  _   _  __| |_ __ ___    ___  ___| |__
| |   | |/ _ \| | | |/ _` | '_ ` _ \ / _ \/ __| '_ \
| |___| | (_) | |_| | (_| | | | | | |  __/\__ \ | | |
\____|_|\___/ \__,_|\__,_|_| |_| |_|\___||___/_| |_|
=======================================================
Cloudmesh Shell

cm> cloud
+--------------------------+-----------+
| cloud                    | active    |
+==========================+===========+
| alamo                    |           |
+--------------------------+-----------+
| aws                      |           |
+--------------------------+-----------+
| azure                    |           |
+--------------------------+-----------+
| dreamhost                |           |
+--------------------------+-----------+
| hp                       |           |
+--------------------------+-----------+
| hp_east                  |           |
+--------------------------+-----------+
| india_eucalyptus         |           |
+--------------------------+-----------+
| india                    |           |
+--------------------------+-----------+
| sierra_eucalyptus        |           |
```

```
+--------------------------+----------+
| sierra                   |          |
+--------------------------+----------+

cm> cloud on india
...
cloud 'india' activated.

cm> flavor india --refresh
...
Refresh time: 0.190665006638
Store time: 0.0578060150146
+--------+------+--------------+---------+-------+--------+----------------------+
| CLOUD  |   id | name         |   vcpus |   ram |   disk | cm_refresh           |
|--------+------+--------------+---------+-------+--------+----------------------|
| india |    1 | m1.tiny      |       1 |   512 |      0 | 2014-08-26T01-15-20Z |
| india |    3 | m1.medium    |       2 |  4096 |     40 | 2014-08-26T01-15-20Z |
| india |    2 | m1.small     |       1 |  2048 |     20 | 2014-08-26T01-15-20Z |
| india |    4 | m1.large     |       4 |  8192 |     40 | 2014-08-26T01-15-20Z |
| india |    7 | m1.memmedium |       1 |  4096 |     20 | 2014-08-26T01-15-20Z |
| india |    6 | m1.memlarge  |       1 |  8192 |     20 | 2014-08-26T01-15-20Z |
+--------+------+--------------+---------+-------+--------+----------------------+
```

Commands without description

This script assumes that you have a key in:

```
$ ~/.ssh/id_rsa.pub
```

Which will be used to log into the VMs and the machines. This key must be uploaded to the FutureSystems portal.

For ubuntu use

```
$ git clone https://github.com/cloudmesh/cloudmesh.git
$ virtualenv ~/ENV
$ source ~/ENV/bin/activate
$ cd cloudmesh
$ sudo ./install system
#
# The command requires input
#
$ ./install requirements
$ ./install new
$ ./install cloudmesh
$ cm-iu user fetch --username=$PORTALNAME
$ cm-iu user create
$ fab mongo.reset
#
# The command requires input
#
$ fab server.start
$ cm cloud list
$ cm cloud on india
$ cm flavor india --refresh
```

For OSX use

```
#
# make sure you installed xcode and do xcode-select --install
#
$ git clone https://github.com/cloudmesh/cloudmesh.git
$ virtualenv ~/ENV
$ source ~/ENV/bin/activate
$ cd cloudmesh
$ ./install system
#
# The command requires input
#
$ ./install requirements
$ ./install new
$ ./install cloudmesh
$ cm-iu user fetch --username=$PORTALNAME
$ cm-iu user create
$ fab mongo.reset
#
# The command requires input
#
$ fab server.start
$ cm cloud list
$ cm cloud on india
$ cm flavor india --refresh
```

One line install with curl

> **Warning:** This method is experimental, please give us feedback.

This script can also be executed while getting it from our convenient instalation script repository. For ubuntu you can use:

```
$ curl -sSL https://cloudmesh.github.io/get/ubuntu/ | username=$PORTALNAME sh
```

It will install cloudmesh in the directory where you started it from and place it in the directory:

```
$ cloudmesh
```

It creates also a directory called *./github/cloudmesh* and then cds into this directory to conduct the installation from there. Furthermore, as you can see this script also creates a virtual env under the name ~/ENV

If you do not like these names or have a conflict with the names, please download the script and modify accordingly.

After you have installed cloudmesh it is important to set a different password for the local cloudmesh user. This is done with:

```
$ cd cloudmesh
$ fab user.mongo
```

Tips

If you lost the cursor on your terminal, you can use the command:

```
$ reset
```

to bring the terminal in its default settings.

7.2.2 Quickstart for an Openstack VM

A video about the contents of this page is available on  rcecpgm-47g.

|video-fs-account|

Note: This setup is primarily used for testing, but it can also be useful for classes using OpenStack, when the call participants have access to an OpenStack cloud.

Setting up Cloudmesh on a VM is an especially convenient way during development and testing. To do so, you can follow the steps to run cloudmesh in a VM running Ubuntu 14.04 on FutureSystems *India* OpenStack. The instructions have been tested on a small instance and the whole process could take about half an hour before you can access the running server.

Requirements

We assume that you have set up an account on FutureSystems and are able to log into the machine with the name india.

If you use a different cloud, you can adapt the instructions accordingly.

Starting the VM

First, you have to start a VM on the cloud and assign it a public IP.

This can be done in multiple ways, using the command line, vagrant, or the horizon GUI. Let us assume you have set it up via the horizon GUI or the novaclient command line.

To set up a mchine you could use the either of the folloing methods:

- **Horizon**: see our manual page.
- **nova**: see the *OpenStack on FutureSystems*
- **cloudmsh**: see *Quickstart on your desktop* and the cloudmesh shell found elsewhere.

Note:

FutureSystems Portalname and Project ID For this example we assume you have set the shell variable PORTALNAME to your FutureSystems portal username. This can be done as follwows. Let us assume your portal name is *albert*. Than you can set it with:

```
export PORTALNAME=albert
```

We also assume that you have a project id that you set to:

```
export PROJECTID=fg101
```

if it is the number 101.

Note: Please note that in the following document we use the $USER and $PORTALNAME are the same values and

the portalname needs to be replaced with the portal name you obtained for FutureSystems. As the subsequent steps are all executed on india we can simply use the default $USER shell variable.

However, we use here a commandline approach and use the tools already installed on india. Thus you do not have to install anything on your machine. We assume however that you have uploaded the public key of your machine to the FutureSystems portal so you can log into india.

We summarize the following steps:

```
$ ssh $PORTALNAME@india.futuresystems.org
india$ module load novaclient
india$ source ~/.futuregrid/openstack_havana/novarc
india$ nova keypair-add --pub-key ~/.ssh/id_rsa.pub $USER-india-key
```

This assumes such a key exists in the location:

```
$ ~/.ssh/id_rsa.pub
```

If you do not have such a key, you can generate it with:

```
$ ssh-keygen -t rsa -C $USER-india-key
```

Remember to set a passphrase once prompted to secure your private key.

Warning: You must not use a passphrase less key! Please specify a strong passphrase.

Next step is to open the necessary ports of the VM to be started:

```
india$ nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0
india$ nova secgroup-add-rule default tcp 22 22 0.0.0.0/0
india$ nova secgroup-add-rule default tcp 8888 8888 0.0.0.0/0
india$ nova secgroup-add-rule default tcp 5000 5000 0.0.0.0/0
india$ nova secgroup-list-rules default
```

Now you cab boot a VM and set public ip for external access:

```
india$ nova boot --flavor m1.small --image "futuregrid/ubuntu-14.04" --key_name $USER-india-key $USE

india$ nova floating-ip-create

india$ export MYIP=`nova floating-ip-list | fgrep "None" | cut -d '|' -f2 | head -1`
india$ nova add-floating-ip $USER-001 $MYIP
india$ nova show $USER-001
```

You should see a table similare like this:

```
+--------------------------------------+-------------------------------------------------------------
| Property                             | Value
+--------------------------------------+-------------------------------------------------------------
| status                               | ACTIVE
| updated                              | 2014-09-12T19:27:30Z
| OS-EXT-STS:task_state                | None
| private network                      | 168.39.1.34, 192.165.159.40
| key_name                             | USER-key
| image                                | futuregrid/ubuntu-14.04 (02cf1545-dd83-493a-986e-583d53ee372
| hostId                               | hsakjfhsdlkjfhsdlkjhflskjdhflkjsdhflkjshfpoeuiyrewuohfkljsdk
| OS-EXT-STS:vm_state                  | active
| OS-SRV-USG:launched_at               | 2014-09-12T19:27:30.000000
| flavor                               | m1.small (2)
```

```
| id                                   | 7e458cbd-d37d-443a-aa76-adc7fcad52ea
| security_groups                      | [{u'name': u'default'}]
| OS-SRV-USG:terminated_at             | None
| user_id                              | sjhkjsahflkjashfkljshfkdjsahfkjh
| name                                 | USER-001
| created                              | 2014-09-12T19:27:23Z
| tenant_id                            | abcd01234hfslkjhfdskjfhkjdshfkjs
| OS-DCF:diskConfig                    | MANUAL
| metadata                             | {}
| os-extended-volumes:volumes_attached | []
| accessIPv4                           |
| accessIPv6                           |
| progress                             | 0
| OS-EXT-STS:power_state               | 1
| OS-EXT-AZ:availability_zone          | nova
| config_drive                         |
+--------------------------------------+-----------------------------------------------------------
```

Looking at the status you will see if the VM is in ACTIVE state. Repeat the command:

```
india$ nova show $USER-001
```

if necessary. Once this is the case you can login to it with:

```
india$ ssh -i ~/.ssh/id_rsa -l ubuntu $MYIP
```

Preparation of the VM

Next you have to update the operating system while logging into the VM:

```
$ sudo apt-get update
$ sudo apt-get install git
```

To obtain cloudmesh you need to clone it from git hub and change to the cloudmesh directory:

```
$ cd ~
$ git clone https://github.com/cloudmesh/cloudmesh.git
$ cd cloudmesh
```

The first thing you have to do is to fix some ip addresses on india with the command:

```
$ ./bin/fix-india-routing.sh
```

Installation

To start the installation of cloudmesh we first need to install a number of packages with:

```
$ ./install system
```

We also recommend that you run virtualenv in python which you can enable with:

```
$ cd ~
$ virtualenv  --no-site-packages ~/ENV
$ source ~/ENV/bin/activate
```

Now let us install cloudmesh into this virtualenv:

```
$ cd cloudmesh
$ ./install requirements
$ ./install new
```

The last command will create a number of yaml files in the folder:

```
$ ~/.cloudmesh
```

Next, install the cloudmesh server anad API with:

```
$ ./install cloudmesh
```

Now we need to populate the cloudmesh.yaml file with your actual information. You can edit the file ‘‘~/.cloudmesh/cloudmesh.yaml‘ either with emacs or vi:

```
$ emacs ~/.cloudmesh/cloudmesh.yaml
```

or:

```
$ vi ~/.cloudmesh/cloudmesh.yaml
```

In this file, update your user profile, name, project data. Alternatively, if you already have yaml files on for example india.FutureSystems.org you can copy your local working yaml files from that machine to the virtual machine.

Yet another alternative is to use the functionality provided by cloudmesh. Before we can use it we have to however create a key that we upload to the FutureSystems portal:

```
$ export PORTALNAME=<put your portal name here>
$ ssh-keygen -t rsa -C $PORTALNAME-ubuntu-vm-key
```

Than lets add the key to the ssh agent:

```
$ eval `ssh-agent -s`
$ ssh-add
```

You will also need to add the key to your FutureSystems portal account. Please visit the portal and past the content of the public key in the appropriate field. YOu can do this by

```
$ cat ~/.ssh/id_rsa.pub
```

Next you can fetch the information you need to acces openstack form india:

```
$ cm-iu user fetch
$ cm-iu user create
```

This will fetch your cloud credentials from FutureSystems and populate them into the yaml config file. We als need to undertake some changes for the india OpenStack cloud configuration with

```
$ fab india.configure
```

To run cloudmesh you will need to start a number of services you will need to create the cloudmesh database. Here we will use the command:

```
$ fab mongo.reset
```

Please note that this command will erase the previous database and you should be carefuly considering its use. When you initialize the cloudmesh server first this is the best method.

Note: Also note that this command will take a long time on machines that do not have SSD’s due to the way mongo sets up the database. Be patient and do not interrup the program although it may run multiple minutes.

Now you are ready to start all services for cloudmesh with:

```
$ fab server.start
```

Then the cloudmesh service should be available via:

```
http://PUBLIC_IP_OF_THE_VM:5000
```

If you forgot your IP, use the command:

```
$ echo $MYIP
```

NOTE:

1. As you might be copying your yaml files into the cloud please secure the VM (following good security practice, including but not limited to proper ssh settings disallowing password authentication, securing the location of your private key as well as setting a passphrase, etc.). As this method targets the scenario for rapid dev and testing, it will be a good idea that shutting the vm down after using.
2. As the server is not secured by HTTPS, remember not to use your favorite password when you are asked to set a password for portal login.
3. This method is only intended for development and testing, and not recommended for real production use.

More information about more sophisticated install instructions are provided at

- http://cloudmesh.futuregrid.org/cloudmesh/developer.html#install-the-requirements

7.2.3 Setup Cloudmesh in an VirtualBox VM with Vagrant for Testing

This tutorial provides as how to deploy Cloudmesh with Vagrant and VirtualBox. Official Ubuntu 14.04 Server LTS 64 bit and 32 bit are supported as base images of Vagrant.

Download cloudmesh

```
$ git clone https://github.com/cloudmesh/cloudmesh.git
$ cd cloudmesh
```

Install Vagrant and VirtualBox

This instructions are tested on Ubuntu 14.04.

```
$ sudo apt-get install vagrant
$ sudo add-apt-repository multiverse
$ sudo apt-get update
$ sudo apt-get install virtualbox
```

Install Veewee (Optional)

There are requirements prior installing Veewee.

```
$ gem install veewee
```

- On OS X Mavericks:

```
$ ARCHFLAGS=-Wno-error=unused-command-line-argument-hard-error-in-future gem install veewee
```

Vagrant up

```
$ cd ~/cloudmesh/vagrant/example1
$ ./run.sh
```

FutureGrid Portal ID

Provide your portal ID:

```
==================================
Futuregrid portal id? (def: )

==================================
```

Base Image

Select one of the base images:

```
==================================
Select base image to launch
==================================
1) Ubuntu Server 14.04 64bit
2) Ubuntu Server 14.04 32bit
Please choose an option:

Ubuntu Server 14.04 xxbit selected
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Checking if box 'ubuntu/trustyxx' is up to date...
```

Vagrant will be loaded.

Vagrant ssh

Cloudmesh installed on a root account. You need to switch user account to a root once you ssh to the VM.

```
$ vagrant ssh
$ sudo su -
```

Virtualenv and cm

Cloudmesh installed on Virtualenv. You need to enable the environment. `cm` Cloudmesh interactive shell is ready to use.

```
$ source ~/ENV/bin/activate
$ cd cloudmesh
$ cm
```

7.2.4 Creating the yaml file

You must have installed cloudmesh as discussed in ??? and run:

```
$ ./install/new
```

This will create a *~/.cloudmesh* directory with some basic yaml files that you will need to modify.

Adding FutureGrid Openstack clouds on sierra and india to the yaml file

For FutureGrid we have additionally provided a script that automatically creates some yaml files from the installation. In future FutureGrid will provide directly a yaml for cloudmesh so that this step is unnecessary. Before you can execute this command you maust make sure that you can log into india and sierra via ssh. Once you have verified this for example with:

```
$ ssh $PORTALNAME@india.futuregrid.org hostname
$ ssh $PORTALNAME@sierra.futuregrid.org hostname
```

Now create the yaml file while fetching some information from the remote machines:

```
$ ./install rc fetch
```

First it will ask you which username you have on FutureGrid. The name may be different from your current local machine name. Please enter your name when you see:

```
Please enter your portal user id [default: albert]:
```

After this you can update the yaml files with the data fetched from the india and sierra with the command:

```
$ ./install rc fill
```

The reason why we have separated the commands and not just created one command is to provide you with the ability to double check overwriting possibly an existing rc file.

Adding FutureGrid OpenStack Clouds on alamo and hotel to the yaml file

We do not recommend adding these machines as they use the FG portal and password. However if you do so, we have placeholders in the yaml file for these clouds. In case you can not find them, simply copy the one from india and make appropriate corrections.

Adding HP cloud to the yaml file

The cloud offered from HP is an Openstack cloud and contains the ability to conduct project and user based billing. As this cloud is Openstack it behaves much the same as the once defined on India and Sierra. There may be differences based on the version.

HP provides an interface to their cloud through horizon. The documentation for it can be found at:

- http://docs.hpcloud.com/hpcloudconsole

To use the cloud you have to first create an account with HP, which will charge you real money for using their cloud. Make sure you understand what costs will be charged before you request thousands of virtual machines. Naturally this is valid for any other commercial cloud also. The console for the HP cloud is available at:

- http://www.hpcloud.com/console

Which will bring you to their horizon interface:

- https://horizon.hpcloud.com

You can add your username and password into the cloudmesh.yaml in the .cloudmesh directory. It is that simple. However, presently the data is stored in cleartext which we will change in future. Thus if your would like to run cloudmesh we currently recommend running it on your own local machine. Make sure that the access to the yaml file is properly secured.

Adding AWS to the yaml file

Amazon EC2 Cloud requires Secret Access Keys to use Amazon Web Services (AWS). To configure Amazon EC2 on Cloudmesh, you need to provide the Secret Access Keys of your account. Amazon allows only to download the credentials via their web page, you need to go to the Security Credentials page to get the credentials. You may use your existing AWS account or create a new AWS account. The Access Key is a pair of Access Key ID and Secret Access Key and these values should be replaced with *EC2_ACCESS_KEY* and *EC2_SECRET_KEY* fields in the yaml file. Cloudmesh identifies *cm_type: aws* as Amazon Web Services in the yaml file, you update the *aws* section with your security credentials. Note that Amazon offers commercial services, the access key identification and the secret key should be kept in a safe place to avoid any unexpected usage from someone who you didn't authorize.

Adding Azure to the yaml file

Microsoft Windows Azure offers security credentials per a valid subscription on a user account. Based on the subscription id, chargeable usage is going to be applied to your bill. To authenticate requests to Azure, you need to configure your credentials for Cloudmesh. The following step-by-step tutorial explains the configuration of Azure credentials on Cloudmesh.

To connect Azure Virtual Machines to Cloudmesh, you need to provide Azure credentials to authenticate requrests in the yaml file. You can find the credentials in the

- Azure Management Portal

which is a web interface to manage your account and Azure Virtual Machines. Also, you can find credentials by downloading the subscription file (.publishsettings) here:

- http://go.microsoft.com/fwlink/?LinkId=254432.

Once you download the file, you may need to import your subscription Id and valid X.509 certificate from the file with the help of the Azure cross-platform command line interface. More information about the Azure CLI can be found in the Manual/article about the

- Azure Cross-Platform Command-Line Interface.

The Azure credentials require that the X.509 certificate is placed in the *.cloudmesh* directory. The *subscriptionid* field should be filled with your Azure subscription id. The valid X.509 certificate file (.pem) must also be stored in the *.cloudmesh* directory. We store it under the name:

```
$HOME/.cloudmesh/azure_managementCertificate.pem
```

Cloudmesh yaml file has an example invalid entry that you can change with your settings. It can be easily identified while looking for the keyword azure in the *cloudmesh.yaml* file. As Azure is a commercial service it is important that you properly secure the .cloudmesh directory and its yaml files.

Note: Recommended files and directory permissions for Secured Cloudmesh To protect the yaml files against any access from other users, we recommend to use *chmod* command. Try *chmod -R o+rwx,go-rwx ~/.cloudmesh* to make any file in the *.cloudmesh* directory a private file to your user account. This way allows you have a full access to the files and the directory but not others.

Azure Quickstart

Azure account If you do not have an Azure account you can obtain one from Microsoft. Microsoft provides a free-trial for new account applicants. The Windows Azure site is located at

- https://manage.windowsazure.com

Download credentials Form ther you can download the:

```
.publishsettings
```

Install Azure CLI Next you will need to install the Azure CLI. This is documented at

- http://azure.microsoft.com/en-us/documentation/articles/xplat-cli/

Here you find install instructions fror Linux but also a link to an OSX installer.

Once the client is installed you can download the credentials

Import Credentials via Azure CLI

```
$ azure account download
$ azure account import <.publishsettings file path>
```

Download Subscription File (.publishsettings)

- http://go.microsoft.com/fwlink/?LinkId=254432

Place X.509 certificate on Cloudmesh

```
$ cp -p ~/.azure/managementCertificate.pem ~/.cloudmesh/azure_managementCertificate.pem

Only the owner with read and write permission e.g. -rw-------
```

Note: Recommended files and directory permissions for Secured Cloudmesh To protect the yaml files against any access from other users, we recommend to use *chmod* command. Try *chmod o+rwx,go-rwx ~/azure_managementCertificate.pem* to make the file a private file to your user account. This way allows you have a full access to the file but not others.

Replace Subscription ID

```
$ azure service cert list
```

provides your subscription id that just imported from the .publishsettings file.

Now, you are ready to use Azure Virtual Machines on Cloudmesh.

Test Azure Virtual Machine TBD

Adding devstack to the yaml file (TBD)

DevStack offers an easy method to try out Openstack on your machine or in a virtual machine (VM). DevStack provides a setup guide and configuration here: Configuration.

Adding dreamhost to the yaml file

Dreamhost provides an Openstack cloud that can be accessed through the dreamhost panel at:

- https://panel.dreamhost.com/index.cgi

The Horizon interface is located at

- https://dashboard.dreamcompute.com

If you are a customer of dreamhost, use your username and password that was send to you.

To use cloudmesh, please add this username and password in the placeholder for dreamhost.

7.2.5 Nosetests

If you would like to verify installation and other features of Cloudmesh, we provide couple of nosetests to make sure that you have working Cloudmesh. These test cases perform several tasks towards Cloudmesh installation, vm creation, termination and others on cm console, cm API, and cm shell.

What does the test involve?

For API, shell, and cm, the nosetests checks activation, list, refresh, start , stop and other features of vm instances. With these tests, we can assure that Cloudmesh users can start or stop vm instances on any interfaces including web gui. The nosetest for the installation does perform actual process of the installation so all the required packages and files will be re-installed and re-configured.

Installation

> **Warning:** your $HOME/.cloudmesh and packages will be re-installed.

Try to run the following command:

```
$ nosetests -v --nocapture ~/cloudmesh/tests/test_cm.py
```

API

Try to run the following command:

```
$ nosetests -v --nocapture ~/cloudmesh/tests/test_cm_api.py
```

cm shell

Try to run the following command:

```
$ nosetests -v --nocapture ~/cloudmesh/tests/test_cm_shell.py
```

cm console

Try to run the following command:

```
$ nosetests -v --nocapture ~/cloudmesh/tests/test_cm_console.py
```

vm test

This nosetest performs 14 tests regarding vm management. It includes:

```
- activation test (`cloud on CLOUDNAME`)
- vm creation (`vm start`)
- vm list (`vm list`)
- vm deletion (`vm delete`)
```

Try to run the following command:

```
$ nosetests -v --nocapture ~/cloudmesh/tests/test_cm_console_ext.py
```

7.3 Cloudmesh cm

7.3.1 Cloudmesh `cm` Command

Notebook

iPython Execution of shell commands

In this section we use one of the build in features of iPython. IPython provides various mechanisms to call programs within its shell. One of the ways to do so is to use the ! character at the beginning of a line to execute the command in the shell.

However, there are more convenient ways to eliminate the ! sign at the beginning of a line. One way is to use the alias command, another is to use the %rehasx command.

IPython Alias

With the alias command we simply define a new command with the name cm that we can call directly from IPython. Here we make sure that the parameters ar between "" so that they are properly set. Just execute the following lines.

```
alias cm cm %s
```

Now let us test the command and lets print the version of cloudmesh cm

```
cm version
```

```
1.0.6
```

Python %rehashx

In addition to the direct specification IPython has also a rehashx function, that loads the commands found in the $PATH variable so you can execute the without !.

```
%rehashx

cm version

1.0.6
```

We are using now one of the methods to call the cm commands in the following sections.

cm Command

The cm command has a number of options that are useful to pass a script or a command directly into cm. Please however not that in some cases the command must be quoted to avoid confusion between flags used for cm and flags used for its subcommands. Let us invoke the -h flag to see which options cm has.

```
cm -h

cm.

    Usage:
      cm [-q] help
      cm [-v] [-b] [--file=SCRIPT] [-i] [COMMAND ...]

    Arguments:
      COMMAND                 A command to be executed

    Options:
      --file=SCRIPT  -f  SCRIPT  Executes the script
      -i                 After start keep the shell interactive,
                         otherwise quit [default: False]
      -b                 surpress the printing of the banner [default: False]
```

Help

Now let us execute the help command to see what other functions are supported. As cm is based on cmd3 that you can find in pypi it inherits a number of commands from cmd3. However, more importantly it also obtains a number of commands from cm itself. To more easily distinguish the categories of the cloud related commands we introduced two of them called GUI commands and cloud commands.

```
cm help

Documented commands (type help <topic>):
========================================
EOF       dot2      help       list      plugins   script          verbose
banner    edit      image      loglevel  project   security_group  version
clear     exec      info       man       py        storm           vm
cloud     exp       init       metric    q         timer           web
color     flavor    inventory  notebook  quit      use             yaml
debug     graphviz  key        open      rain      user
defaults  group     label      pause     register  var

Ipython Commands
================
notebook

Gui Commands
============
```

```
web

Cloud Commands
==============
cloud     group  inventory  rain            storm  yaml      keys
defaults  image  list       register        user   debug     project
flavor    init   metric     security_group  vm     loglevel
```

Starting the Web Browser

To start the browser, simply type the command

Listing Clouds

```
cm cloud list
```

```
+-----------+--------+
| cloud     | active |
+-----------+--------+
| aws       |        |
| azure     |        |
| devstack  |        |
| dreamhost |        |
| hp        |        |
| hp_east   |        |
| india     | True   |
+-----------+--------+
```

Let us inspect the parameters. To limit the output we just display the first 10 lines of the help/man page. We see the –column option in the list command.

```
cm help cloud | head -n 10
```

```
::

    Usage:
        cloud [list] [--column=COLUMN] [--format=FORMAT]
        cloud info [CLOUD|--all] [--format=FORMAT]
        cloud alias NAME [CLOUD]
        cloud select [CLOUD]
        cloud on [CLOUD]
        cloud off [CLOUD]
```

...

For more information, read the help page. It essentially allows us to display some more useful information beyond to just document the active clouds. Let us also display the label. This is done with the following command.

```
cm "cloud list --column=active,label"
```

```
+-----------+--------+--------------------+
| cloud     | active | label              |
+-----------+--------+--------------------+
| aws       |        | aws                |
| azure     |        | waz                |
| devstack  |        | localhost_icehouse |
```

```
| dreamhost |        | dreamhost          |
| hp        |        | hpos               |
| hp_east   |        | hpeos              |
| india     | True   | ios_havana         |
+-----------+--------+--------------------+
```

Let us now demonstrate a common error by not using proper quoting. This occurs when you use option flags with the command. Here our current parser is unable to distinguish between the options passed to cm and the options as used in the cm command. You see the usage message that we do not have a –column in the cm command. To avoid thie use the " " as previously shown.

```
cm cloud list --column=active,label

Usage:
      cm [-q] help
      cm [-v] [-b] [--file=SCRIPT] [-i] [COMMAND ...]
```

7.3.2 `project` Command in Cloudmesh `cm`

Notebook

The `project` command provides a list of default, active, or completed project for a given user. You can update the project information with this command.

Note: that all your `project` command executions update your yaml

file (cloudmesh.yaml) and mongo db (user, defaults) both.

IPython Alias

With the alias command we simply define a new command with the name cm that we can call directly from IPython. Here we make sure that the parameters ar between "" so that they are properly set. Just execute the following lines.

```
alias cm cm %s
```

We are using now one of the methods to call the cm commands in the following sections.

`project info` Command

The `project` command has a number of options that are useful to manage a list of projects. The `info` option provides a current information on your account.

```
cm project info

Project Information
-------------------

   default: fg2
  projects: fg2, fg82, fg415
 completed: fg1003
```

Set a default project

`project default [project name]` allows you to update a default project to your account.

```
cm project default fg2

fg2 project is a default project now

cm project

Project Information
-------------------

   default: fg2
  projects: fg2, fg82, fg415
 completed: fg1003
```

You can see your default project has been changed from fg20 to fg82. Note that `cm project` is same command with `cm project info`. `info` option can be suppressed.

Set an active project

You can add a project to your account as one of the active projects. You can have multiple projects in your active project list.

```
cm project active fg415

fg415 project is an active project(s) now

cm project info

Project Information
-------------------

   default: fg2
  projects: fg2, fg82, fg415
 completed: fg1003
```

You can see the new project `fg415` added to the active projects.

Change the status of project to `completed`

If your project is completed and no longer needed, you may want to change the status from active to completed. `completed` option allows to change the status of the selected project.

```
cm project completed fg415

fg415 project is in a completed project(s)

cm project

Project Information
-------------------

   default: fg2
```

```
  projects: fg2, fg82
 completed: fg1003, fg415
```

Add a project

You can use three commands to add a project. `cm project active` allows you to add a project to the active list. `cm project default` sets a default project. `cm project completed` adds a project to the completed list. For example, you can add fg999 to the active like below.

```
cm project active fg999

fg999 project is an active project(s) now
```

Delete a project

`project delete` command simply performs the deletion of the given project in the yaml file.

```
cm project delete fg999

fg999 project is deleted
```

Help message

```
cm "project -h"

Usage:
       project
       project info [--json]
       project default NAME
       project active NAME
       project delete NAME
       project completed NAME

Manages the project

Arguments:

  NAME           The project id

Options:

   -v       verbose mode

Usage:
       project
       project info [--json]
       project default NAME
       project active NAME
       project delete NAME
       project completed NAME

Manages the project
```

```
Arguments:

  NAME           The project id

Options:

   -v       verbose mode
```

7.3.3 Command Shell Output Control

Notebook

The `debug` command provides to turn on the debug log level or to turn it off.

IPython Alias

With the alias command we simply define a new command with the name cm that we can call directly from IPython. Here we make sure that the parameters ar between "" so that they are properly set. Just execute the following lines.

```
alias cm cm %s
```

`debug` Command in Cloudmesh `cm`

The `debug on` command changes the log level to *debug*. In *debug* mode, all log messages will print out on the screen.

```
cm debug on
```

```
cm debug off
```

`loglevel` Command in Cloudmesh cm

`loglevel` allows you to update a log level on Cloudmesh. We can chose one of five log levels: * `DEBUG` * `INFO` * `WARNING` * `ERROR` * `CRITICAL`

To view the current log level you can use the command:

```
cm loglevel
```

To set the log level you simply put the keywords debug, info, warning, error, or critical after the loglevel comamnd:

```
cm loglevel debug
```

`yaml` Command in Cloudmesh cm

Update a value in a yaml file

You can change a value on a selected yaml file on the Cloudmesh cm shell. You may need to restart the Cloudmesh server to reflect the change. `yaml info` provides a current information of the yaml file (cloudmesh.yaml).

```
cm yaml info
```

View the value of the selected key in the server yaml file

`yaml info-server` provides information about the server yaml file (cloudmesh_server.yaml). If you specify the key name in a dotted format, you can see the stored value in the yaml file. This example below selects *['cloudmesh']['server']['loglevel']* in the yaml file.

```
cm yaml info-server cloudmesh.server.loglevel
```

You can see the *DEBUG* is the current setting of the log level in the Cloudmesh server.

Change the value of the selected key

If you desire to change the log level to *ERROR*, you need to call `replace-server` option in the yaml command.

```
cm yaml replace-server cloudmesh.server.loglevel ERROR
```

```
cm yaml info-server cloudmesh.server.loglevel
```

Now you can see it has been changed to *ERROR*.

`color` Command in Cloudmesh cm

The Cloudmesh cm shell can enable color support in output messages. Warning, Error, Debug, Info, or other type of messages can be viewed with different colors. `cm color on/off` command enables or disables this feature.

```
cm color on
```

```
cm color off
```

7.3.4 CM Console for VM Management

Notebook

```
alias cm cm %s
```

The alias command is only needed when executing the commands in this in IPython.

First let us set the error reporting to a minimal. If you like to see more output you can switch it on by setting different debug and loglevels.

Preparation

```
cm debug off
```

```
cm loglevel error
```

Lets get the username

```
cm user id
```

Help

Let us review the available commands

```
cm help
```

Activating Clouds

In order for cloudmesh to work with multiple clouds, we need to find out first which clouds are available. Users can add their own clouds later which we describe in the registration section.

Let us inspect which clouds are available by invoking the list command.

```
cm cloud list
```

As you see we have a number of clouds, but none of them is already active. Thus we need to first activate a cloud. We assume that you have an account on FutureGrid. Let us activate the cloud india

```
cm cloud on india
```

We also have a conveniet interactive command to select a cloud to work with, that however does not work with ipython

```
"cloud select"
```

or you may also input "cloud select india" to select a specific cloud india.

To check if the cloud was activated, simply use the list command again.

```
cm cloud list
```

Start a VM

Now let us see how to start a VM on a cloud, here is how to start a VM on cloud india.

```
cm "vm start --cloud=india --image=futuregrid/ubuntu-14.04 --flavor=m1.small"
```

Set a default flavor or image

Each cloud must have a default image and a default flavor to launch vm instances in a simple step. The `cloud set` command provides a way to set default values for an image or a flavor.

```
cm "cloud set flavor india --flavor=m1.small"
```

```
cm "cloud set image india --image=futuregrid/ubuntu-14.04"
```

Get Flavors or Images

Available flavors can be listed with the following command.

```
cm "list flavor india"
```

refresh option updates the data from the IaaS cloud. The cached data in the mongo database will be updated.

```
cm "list flavor india --refresh"
```

The `list image` command provides an available vm images on a selected cloud.

```
cm "list image india --column=name,updated"
```

Quick Start a VM

The `vm start` command provides a quick launch of a vm instance in cloudmesh.

We use `vmname` to delete the vm instance later in this tutorial.

```
_vmname=!cm "label --raw"
vmname=_vmname[0]

cm "vm start --name={vmname} --cloud=india"
```

Delete a VM

If you know the id or name of the virtual machine that you want to destroy, *delete* command in cloudmesh simply terminate the instance.

```
cm "vm delete {vmname} --cloud=india --force"
```

More options to launch a VM instance

When you create a new VM instance, you can also choose multiple options such as a flavor, an image associated with the instance. The `vm start` command accepts optional parameters as a user input of these options. To see a brief description of the command, try `cm "vm --help"` in the IPython Notebook cell.

Available options are:

- `–cloud=<CloudName>` : give a cloud to work on, if not given, selected or default cloud will be used
- `–count=<count>` : give the number of servers to start
- `–flavor=<flavorName>` : give the name of the flavor
- `–flavorid=<flavorId>` : give the id of the flavor
- `–group=<group>` : give the group name of server
- `–image=<imgName>` : give the name of the image
- `–imageid=<imgId>` : give the id of the image

```
cm "vm start --cloud=india --flavor=m1.medium --image=futuregrid/ubuntu-12.04"
```

Set default cloud

If you want to make things even more convenient, you can set a default cloud or select a cloud to work with so that you don't have to type in a cloud everytime you need to specify a cloud, to set india as default cloud

```
cm "cloud set default india"
```

to select a cloud

```
cm "cloud select india"
```

You can see a selected cloud as a temporarily default cloud to work with.

For more details of using command cloud to set up a cloud

cm "cloud -h"

Refreshing VM status

After you have started or deleted VMs, you may want to check clouds' VMs status. To refresh cloud india's VMs information

```
cm "list vm india --refresh"
```

Starting multiple VMs

Sometimes we want to start more than one VM at the same time, we can choose the option –count=int where int is the number of VMs you want to start. For example, to start 3 VMs on india

```
import uuid;
temp_group_name="ipython-tutorial-cm-" + str(uuid.uuid4().get_hex().upper()[0:6])
```

We use the temporary group name to delete the VMs later in this tutorial.

```
cm "vm start --cloud=india --count=3 --group={temp_group_name}"
```

Deleting VMs

To delete one VM is easy, what if we want to delete 1000 VMs, we need a more convenient way to do it. Cloudmesh shell provides several methods to find the VMs and delete them, you may think there are two phases of VM deletion, searching and deleting. Here are some examples:

Let's start with deleting the VMs that we started in the previous example. We can delete them with the group name that we used. `temp_group_name`

```
cm "vm delete --cloud=india --group={temp_group_name} --force"
```

to delete all VMs of cloud india

```
cm "vm delete --cloud=india --force"
```

Note here we use the option "–force", without it the shell will give you a list of VMs to delete and ask for your confirmation.

to delete a VM by giving its name (you may always provide a cloud unless you have specified a default cloud or have selected a cloud)

```
cm "vm delete --cloud=india sample_vm --force"
```

to delete a VM by group

```
cm "vm delete --cloud=india --goup=testgroup --force"
```

We can also narrow the search result by giving more search conditions. For example, to delete VMs of cloud india that they are also in the group 'testgroup' and they have the prefix name 'abc' and their indices' range is no greater than 100

```
cm "vm delete --cloud=india --goup=testgroup --prefix=sample --range=,100
--force"
```

Login to a VM

To login to a VM, you need to assign a public id to the VM you started, for example, assign a public ip to a VM named test_1 on cloud india:

```
_vmname=!cm "label --raw"
vmname=_vmname[0]

cm "vm start --cloud=india"

cm "vm ip {vmname} --cloud=india"

cm "vm ip test_1 --cloud=india"
```

Then you can login to this VM by(note here you need to provide the login name for your VM, it varys depend on the image you use while you start the VM, e.g. for ubuntu, you may type –ln=ubuntu):

```
cm "vm login test_1 --ln=ubuntu --cloud=india"
```

If you just want to run some commands and get the return on the VM, you may add the commands at the end following '–'(e.g. ls -a)

```
cm "vm login {vmname} --ln=ubuntu --cloud=india -- ls -a"
```

For more details for command vm

```
cm "vm -h"
```

```
cm "vm delete {vmname} --cloud=india"
```

Stop the vm that we started for this tutorial.

Vitual Machine Name

In Cloudmesh, the default name of VM consists of your username and a number, for example, alex_1. `label` command allows you to manage or modify the VM name as you wish.

```
cm label
```

```
cm "label --prefix=gregor --id=40"
```

If the user doesn't provide a name while starting VMs, cloudmesh will generate labels for them. The default form to name VMs is prefix_index, where prefix is a string and index is an non-negative integer. If a index is used, the index value will be automatically added by one waiting to be used for next VM. To check your current prefix and index

7.3.5 Metric Commands in Cloudmesh `cm`

Notebook

The `limits` command provides active usage data with quota sizes. Resource utilization can be viewed with this command. For more details, `quota` command shows all quota information for a project, `usage` command provides a usage report for a certain period.

> **Warning:** `limits` command is only supported in OpenStack

IPython Alias

With the alias command we simply define a new command with the name cm that we can call directly from IPython. Here we make sure that the parameters ar between "" so that they are properly set. Just execute the following lines.

```
alias cm cm %s
```

We are using now one of the methods to call the cm commands in the following sections.

`limits` command

Simple execution of `limits` command displays vCPU cores, floating ip addresses (public ip), memory sizes (in megabytes), security groups, and vm instances. Other usage data suppressed and can be observed by `usage` or `quota` command.

```
cm limits

* india
Refreshing albert servers india ->
Refresh time: 0.157671928406
Store time: 0.00192499160767
+----------------+--------------+
| Limits         | (Used/Max)   |
+----------------+--------------+
| Cores          | 3 / 20       |
| FloatingIps    | 3 / 10       |
| RAM            | 6144 / 51200 |
| SecurityGroups | 5 / 10       |
| Instances      | 3 / 10       |
+----------------+--------------+
```

`usage` command

`usage` command provides a short report for usage data of your cloud platform. You can choose desired start and end dates of your report.

```
cm usage

* india
Refreshing albert servers india ->
Refresh time: 0.178751945496
Store time: 0.00191593170166
+----------------------+----------------------------------+
| Variable             | Value                            |
+----------------------+----------------------------------+
| total_memory_mb_usage | 368384.373737                   |
| total_vcpus_usage     | 179.875182489                   |
| total_hours           | 179.875182489                   |
| tenant_id             | 3e6eaf1d913a48f694a7bc0fbb027507 |
| stop                  | 2014-11-20T02:38:19.747782      |
| server_usages         | 11 vms                          |
| start                 | 2014-11-19T02:38:19.747765      |
| total_local_gb_usage  | 3597.50364978                   |
+----------------------+----------------------------------+
```

start and stop rows indicate the period of this report. In this example, the usage value displays 1 day usage data between 11/19/2014 - 11/20/2014.

`quota` command

`quota` command shows limit policy on your resource usage towards your project.

```
cm quota

* india
Refreshing albert servers india ->
Refresh time: 0.152929067612
Store time: 0.00176787376404
+------------------------------+----------------------------------+
| Variable                     | Value                            |
+------------------------------+----------------------------------+
| injected_file_content_bytes | 10240                            |
| metadata_items              | 128                              |
| ram                         | 51200                            |
| floating_ips                | 10                               |
| key_pairs                   | 100                              |
| injected_file_path_bytes    | 255                              |
| instances                   | 10                               |
| security_group_rules        | 20                               |
| injected_files              | 5                                |
| cores                       | 20                               |
| fixed_ips                   | -1                               |
| id                          | 3e6eaf1d913a48f694a7bc0fbb027507 |
| security_groups             | 10                               |
+------------------------------+----------------------------------+
```

These quota information tells you how many resources you can use on this account.

7.3.6 Deploying Virtual Cluster with Cloudmesh `cm`

Notebook

To support the creation of a virtual compute cluster (e.g. Hadoop or Slurm) on cloudmesh, we are offering the command `cluster` to start, configure, manage or update a number of virtual machines.

IPython Alias

With the alias command we simply define a new command with the name cm that we can call directly from IPython. Here we make sure that the parameters ar between “” so that they are properly set. Just execute the following lines. If you do not use ipythin, you can skip this step.

```
alias cm cm %s
```

Preparation (Optional)

The `cluster` command is available in the cloudmesh shell. There are certain steps to make sure you are all set to deploy a new cluster. You can SKIP these steps if you have configured cloudmesh previously.

```
cm "cluster"
```

Before starting a cluster, please make sure you activate a cloud:

```
cm "cloud on india"
```

Activation however does not select the cloud as the default cloud to start virtual machines. This is achieved by the select command

```
cm "cloud select india"

cloud default india
```

Default keypair

To gain access to the virtual machines, you need to have a key registered with the cloud. If you don't have a default keypair, you need to set or need to specify which keypair you are going use for the vm.

```
cm "key default test-key"
```

VM Name

You can also define the name of VMs which contains a prefix and an index with the `label` command. However, the name of the vms must be unique. By default your username and a number will be used that will be automatically increased. Thus we recommend that you avoid using this command.

```
cm "label --prefix=test --id=1"
```

Default Image

You can choose a default image to create a virtual machine. In this example, we use ubuntu-14.04 image as a default.

```
cm "default image --name=futuregrid/ubuntu-14.04"
```

Default Flavor

You can chose a default flavor. However, make sure that the falvor actially works with the specified image. Some images require a minimal flavor.

```
cm "default flavor --name=m1.small"
```

Create Cluster

Then you may start the cluster with command 'cluster create' by providing the following values:

`--count`
: specify amount of VMs in the cluster

`--group`
: specify a group name of the cluster, make sure it's UNIQUE

`--ln`
: login name for VMs, e.g. ubuntu

Create 3 virtual machines

```
Let us create a cluster with the group name test that contains 3 virtual machines.

cm "cluster create --count=3 --group=test --ln=ubuntu"
```

You may also provide a cloud name, flavor or image as parameter in the command if you do not want to use the defaults. For example you can use the following do do so:

```
cm "cluster create --count=3 --group=test0 --ln=ubuntu --cloud=india --flavor=m1.small --image=future
```

You can display the status of vms for the cluster with the `vm list` command:

```
cm "vm list --refresh --group=test"
```

7.3.7 Managing multiple VMs with Cloudmesh cm

When you start a cluster of VMs in Cloudmesh CLI, it is required to specify a group name for the cluster, so that we can manage the cluster according to the cluster name. In Cloudmesh CLI, you may use group command to manage groups.

IPython Alias

With the alias command we simply define a new command with the name cm that we can call directly from IPython. Here we make sure that the parameters are between "" so that they are properly set. Just execute the following lines.

```
alias cm cm %s
```

We are using now one of the methods to call the cm commands in the following sections.

`group` command

The `group` allows us to perform simple grouping of VMs but also images and flavors. A group is essentially a tuple in which you can add to a named group a type of an object and its unique name. The group command allows you to create a group, add items to groups, and list groups. For more information, one can refer to the manual page.

```
cm "group -h"
```

To create a group with the name sample-group you can use the command:

```
cm "group create sample-group"
```

Then you can list the groups with:

```
cm "group list"
```

To add a VM named sample-vm to the group (make sure the VM exists before add it), you can use:

```
cm "group add item sample-group vm sample-vm"
```

To list the elements in a group you can use:

```
cm "group show sample-group"
```

To delete a VM from a group you can use:

```
cm "group remove item sample-group vm sample-vm"
```

To delete a group:

```
cm "group remove sample-group"
```

Please note that deletion of a group does not delete the accociated objects in that group. It only removes the object that stores the types and objects for that group.

Working with group

Groups are conveniently integrated into the cloudmesh shell. You may include a VM into a group while you create it, delete VMs by groups and when you create a cluster, a group name has to be provided.

To start a VM and add it to group, you can use the group parameter. If the group doesn't exist, one will be created:

```
cm "vm start --group=sample"
```

To delete the virtual machines specified in a group you can use the command:

```
cm "vm delete --group=sample"
```

To create multiple vms that are part of a vitual cluster, you need to provide a group name. If the group doesn't exist, one will be created. If the group exist and it has VMs in it already, the newly created virtual machines will be included into the cluster. The following example will create two virtual machines while assuming the default cloud, flavor, and image. Make sure you set them before if they have not been set.

```
cm "cluster create --count=2 --group=sample"
```

7.3.8 Managing Volumes with Cloudmesh `cm`

Cloudmesh provides a command for managing volumes.

IPython Alias

With the alias command we simply define a new command with the name cm that we can call directly from IPython. Here we make sure that the parameters are between "" so that they are properly set. Just execute the following lines.

```
alias cm cm %s
```

We are using now one of the methods to call the cm commands in the following sections.

Before using the volume command make sure to load the configurations by choosing a cloud using nova command, e.g.:

```
cm "nova set india"

india is set
```

for more detailed descriptions:

```
cm "volume"

Error: Wrong Format

Usage:
    volume list
    volume create <size>
                  [--snapshot-id=<snapshot-id>]
                  [--image-id=<image-id>]
                  [--display-name=<display-name>]
```

```
                [--display-description=<display-description>]
                [--volume-type=<volume-type>]
                [--availability-zone=<availability-zone>]
    volume delete <volume>
    volume attach <server> <volume> <device>
    volume detach <server> <volume>
    volume show <volume>
    volume snapshot-list
    volume snapshot-create <volume-id>
                          [--force]
                          [--display-name=<display-name>]
                          [--display-description=<display-description>]
    volume snapshot-delete <snapshot>
    volume snapshot-show <snapshot>
    volume help


volume management

Arguments:
    <size>            Size of volume in GB
    <volume>          Name or ID of the volume to delete
    <volume-id>       ID of the volume to snapshot
    <server>          Name or ID of server(VM).
    <device>          Name of the device e.g. /dev/vdb. Use "auto" for
                      autoassign (if supported)
    <snapshot>        Name or ID of the snapshot

Options:
    --snapshot-id <snapshot-id>
                           Optional snapshot id to create the volume from.
                           (Default=None)
    --image-id <image-id>
                           Optional image id to create the volume from.
                           (Default=None)
    --display-name <display-name>
                           Optional volume name. (Default=None)
    --display-description <display-description>
                           Optional volume description. (Default=None)
    --volume-type <volume-type>
                           Optional volume type. (Default=None)
    --availability-zone <availability-zone>
                           Optional Availability Zone for volume. (Default=None)
    --force                Optional flag to indicate whether to snapshot a volume
                           even if its attached to an instance. (Default=False)

Description:
    volume list
        List all the volumes
    volume create <size> [options...]
        Add a new volume
    volume delete <volume>
        Remove a volume
    volume attach <server> <volume> <device>
        Attach a volume to a server
    volume-detach <server> <volume>
        Detach a volume from a server
    volume show <volume>
```

```
    Show details about a volume
volume snapshot-list
    List all the snapshots
volume snapshot-create <volume-id> [options...]
    Add a new snapshot
volume snapshot-delete <snapshot>
    Remove a snapshot
volume-snapshot-show <snapshot>
    Show details about a snapshot
volume help
    Prints the nova manual
```

7.3.9 Install IPython Server

Cloudmesh contains a convenient mechanism to set up an IPython web server. The server will run on port 8888 of your vm. It will install in your environment a notebook directory in ~/notebook. Here you can place your ipython notebooks. initially this directory will be empty.

You create the notebook server with:

```
cm notebook create
```

You start the notebook server with

```
cm notebook start
```

You can terminate the notebook server with:

```
cm notebook kill
```

When terminating, make sure you saved your notebooks. You can restart the server in case you need to run it again.

7.4 Cloudmesh API

7.4.1 Cloudmesh API Initialization

Notebook

To access the many useful functions available in cloudmesh, we need to import the cloudmesh module.

```
import cloudmesh
```

Version

Cloudmesh has a version number that can be retrived with the version function

```
print cloudmesh.version()
```

```
1.0
```

7.4.2 Defining Names

Notebook

Often we need to define some unique names to distinguish virtual machines or other objects that we use as part of our programming. Besisdes uuid that is provided by python, cloudmesh has additional functions that simplify name generation.

UUIDs

UUIDs are provided as part of the python standard libraries. There are a number of different initializations and which to use depends on your needs.

```
import uuid
```

Machine dependent uuid

One of them is to create UUIDs dependent on your machine name with uuid1.

```
uuid.uuid1()
```

```
UUID('c1888a4a-44c0-11e4-8c4a-600308a5f9d2')
```

Random Uuid

uuid4 creates random uuids.

```
uuid.uuid4()
```

```
UUID('cefa66c8-d8a0-4ded-9bf7-9ab3d178dc43')
```

As in some cases we want to generate names that do not include special characters such as - or . we avoid using the uuid function for now and use the function get_unique_name instaed.

```
uuid.UUID(bytes=uuid.uuid4().bytes)
```

```
UUID('ee73224b-3ee7-426b-b3f1-92906afccf66')
```

```
uuid.uuid4().int
```

```
128265862276225227802881754516275727500L
```

Cloudmesh get_unique_name

Sometimes it is beneficial to create uuids without the - in it. For this we have a convenience function in cloudmesh.

```
from cloudmesh_common.util import get_unique_name
```

```
print get_unique_name()
```

```
c344f40544c011e488c2600308a5f9d2
```

As you can see it is just like the uuid function (currently uuid1) with the - removed.

In addition one can place a prefix into the uuid to make furher distinctions. However this is rarely needed.

```
get_unique_name("gregor")
```

```
'gregorc3ed480244c011e4a547600308a5f9d2'
```

Generating VM names

To create a name for a virtual machine we often recommend to use a prefix with a number for an identifier of the virtual machine. This will come in handy when we need to start multile virtual machines and distinguish them with a different name. Cloudmesh uses internally by using your username from the FutureGrid portal as defined in the cloudmesh.yaml file.

```
import cloudmesh

print cloudmesh.vm_name("gregor", 1)
```

```
gregor-00001
```

Getting the next vm name

Here is how to get the next vm name by using Cloudmesh API, this method gets the data from database. Before call vmname(), we need to get the username and activate the user account.

```
mesh = cloudmesh.mesh("mongo")

username = cloudmesh.load().username()

mesh.activate(username)

mesh.vmname()
```

```
u'TBD_1'
```

7.4.3 Load Command

Notebook

Cloudmesh comes with a number of easy to use configuration files. All of them are yaml files. These configuration files are used to initialize cloudmesh in standalone mode. Hence, they are important when cloudmesh starts up or is run in user more. In server mode the information is typically retrieved from the cloudmesh database.

We focus here on some of the most important configuration files. After deployment they can be found in the `~/.cloudmesh` directory

API for cloudmesh.yaml

Information about the user and which clouds he has access to are stored in the `cloudmesh.yaml` file.

```
import cloudmesh
```

You can load this file with the load command, while either specifying no parameters or using the `"user"` string as parameter.

```
user = cloudmesh.load()
```

Once you have loaded it you can inspect the available clouds. Note that you must have provided credential information for these clouds in the cloudmesh.yaml file prior to starting the program. You also will need to activate a cloud befor eyou can use them. This is discussed elsewhere.

```
user.cloudnames()
```

```
['aws', 'azure', 'devstack', 'dreamhost', 'hp', 'hp_east', 'india']
```

In addition to information about clouds you will also find some information about yourself. Make sure You have specified the information in the yaml file.

```
user.firstname
```

```
'TBD'
```

```
user.lastname
```

```
'TBD'
```

Note: If you see here TBD you need to edit the

`~/.cloudmesh/cloudmesh.yaml` file to add first and lastname.

To obtain the futuregrid username use the command `user.username()`

API for cloudmesh_server.yaml

To configure the server and the databases the server yaml file is used. via the load command it will be loaded into a dict. A special get function can get sub dictionaries.

```
config = cloudmesh.load("server")
```

```
print config.keys()
```

```
['kind', 'meta', 'cloudmesh']
```

```
config.get('meta').keys()
```

```
['yaml_version', 'kind', 'filename', 'location', 'prefix']
```

```
config.get('meta.filename')
```

```
'/Users/flat/.cloudmesh/cloudmesh_server.yaml'
```

```
print config.get('cloudmesh').keys()
```

```
['server']
```

API for cloudmesh_launcher.yaml

We are currently working on integrating PaaS launchers into cloudmesh that easily deploy software based on configuration parameters specified in the launchers. The specification of the launchers are stored under `cloudmesh.launcher.recipies`. We provide the information for such a launcher as an example.

Note: This is a new feature and is not yet fully developed.

```
config = cloudmesh.load("launcher")

print config.keys()

['kind', 'meta', 'cloudmesh']

config.get('cloudmesh').keys()

['launcher']

config.get('cloudmesh.launcher').keys()

['recipies']

config.get('cloudmesh.launcher.recipies').keys()

['mooc', 'slurm', 'hadoop', 'ganglia', 'nagios']

from pprint import pprint
pprint (config.get('cloudmesh.launcher.recipies.mooc'))

OrderedDict([(`name', `Mooc'), (`description', `Deploys a Slurm cluster. One of the Vms is
```

7.4.4 Batch Queues

Notebook

Cloudmesh includes a convenient interface to PBS qstat. It is using an ssh command to connect to the login node on which to execute qstat. The result of the call is available as a dict and you can print it to inspect it closer.

In addition cloudmesh has a customization that alloes resources in FUturegrid to be seperated by machine. This is necessary as the login node on india is also in control of the queues on a number of machines instead of just one.

The username for FutureGrid is loaded from the configuration yaml file in the example

```
import cloudmesh

username = cloudmesh.load().username()

india = cloudmesh.PBS(username, "india.futuregrid.org")
```

QStat

```
qstat = india.qstat()

---------------------------------------------------------------------------
RuntimeError                              Traceback (most recent call last)

<ipython-input-7-cfa496304576> in <module>()
----> 1 qstat = india.qstat()


/Users/flat/ENV/lib/python2.7/site-packages/cloudmesh-1.0-py2.7.egg/cloudmesh/pbs/pbs.pyc in qstat(se
```

```
    285             except:
    286                 raise RuntimeError(
--> 287                     "can not execute pbs qstat on host {0}".format(self.host))
    288             info = {}
    289

RuntimeError: can not execute pbs qstat on host india.futuregrid.org
```

Next, let us list the nuber of clusters managed with the india PBS deployment

```
qstat.keys()
```

```
qstat["india.futuregrid.org"].keys()
```

```
for jobname in qstat["india.futuregrid.org"]:
    job = qstat["india.futuregrid.org"][jobname]
    print jobname, job["job_state"]
```

QInfo

```
qinfo = india.qinfo()
```

To list the queue names, simply print the keys.

```
print qinfo["india.futuregrid.org"].keys()
```

More information is available when you inspect the dict.

7.4.5 Cloudmesh API for VM Management

Notebook

Cloudmesh supports the simple management of heterogeneous virtual machines from a variety of cloud frameworks, including Openstack, Azure, AWS, Eucalyptus and but also AWS compatible clouds. This page shows how to use Cloudmesh functions in python by a couple of examples on starting or stopping virtual machine instances through the APIs.

Initialization

A simple import allows you to enable all features of Cloudmesh in Python.

```
import cloudmesh
```

```
from pprint import pprint
```

Activating Clouds

Cloudmesh provides *yaml* or *mongo* option where to load basic information. *yaml* relies on the yaml files in the $HOME/.cloudmesh directory, *mongo* retrieves information from the mongo database.

```
mesh = cloudmesh.mesh("mongo")
```

Get a username

In most Cloudmesh functions, you need to provide a username to tell the server who is going to use cloud services.

```
username = cloudmesh.load().username()

print username

gvonlasz
```

Activate the user account

With the activation, the connection to IaaS cloud is established.

```
mesh.activate(username)

* india
Refreshing gvonlasz servers india ->
Refresh time: 0.305170059204
Store time: 0.00300216674805
```

Register a cloud

```
cloudmesh.shell("cloud on india")

* india
Refreshing gvonlasz servers india ->
Refresh time: 0.282967090607
Store time: 0.00245690345764
[32mcloud 'india' activated.[0m

cloudmesh.shell("cloud list")

+-----------+--------+
| cloud     | active |
+-----------+--------+
| aws       |        |
| azure     |        |
| devstack  |        |
| dreamhost |        |
| hp        |        |
| hp_east   |        |
| india     | True   |
+-----------+--------+
```

Each cloud must have a default image and a default flavor to launch vm instances in a simple step. default function provides a way to set default values for an image o a flavor. In this example, we use *image*, *flavor* variables which created from the examples above.

Need a help for the function?

You can execute a cell with a function name and a single question mark (?) to see a short description of a function. A double question marks (??) provide a source code with a docstring of the function. Try *mesh.default?* or *mesh.default??*

Get Flavors or Images

Cloudmesh retrieves information from a cache that need to be refreshed in case you like to get the newest values. At the beginning you have to call refresh so that he information from the cloud is populated into the cache.

```
mesh.refresh(username,types=['flavors', 'images'],names=["india"])
```

```
* india
Refreshing gvonlasz flavors india ->
Refresh time: 0.179011821747
Store time: 0.00220489501953
Refreshing gvonlasz images india ->
Refresh time: 0.43066906929
Store time: 0.00897192955017
```

Available flavors can be listed with the following function.

```
flavors = mesh.flavors(cm_user_id=username, clouds=["india"])
```

Let us display the names of the flavors

The functions about vm images provide an available vm images on a selected cloud.

```
for id in flavors["india"]:
    print flavors["india"][id]["name"]
```

```
m1.tiny
m1.medium
m1.small
m1.xlarge
m1.large
```

```
images = mesh.images(clouds=['india'],cm_user_id=username)
```

```
for id in images["india"]:
    print images["india"][id]["name"]
```

```
salsahpc/cloud-mooc-m1-large-4GB
CentOS6
SL64-blank-sparse10gb-C vmdk
cglmoocs/ipython
futuregrid/centos-6
DaLiAna-vm2014e-geant4.10.vmdk Jan best
ubuntu-13.10
futuregrid/ubuntu-14.04
fg101/richieriee/my-ubuntu-01
sl64-gluex-vm2014e-40gb.vmdk Justin
futuregrid/fedora-19
fg10/jcharcal/centos6.5_x86_64
balewski/kernel-2.6.32-431.5.1-sl65
SL64-blank-sparse40GB vmdk
Ubuntu-12.04-blank2.vmdk
balewski/ramdisk-2.6.32-431.5.1-sl65
sl6_x64-qemu  french ?bad
balewski/sl6.5-blank-80gb-b works
grp17Cent
futuregrid/ubuntu-12.04
ndssl-vt/ubuntu-12.04-small
futuregrid/fedora-20
```

```
fg7/rynge/centos6-v1
ubuntu12-comet
balewski/daliana-vm2014e2-sl6.5-geant4.10-root5.34
DaLiAna-vm2014d-SL64-A.vmdk
```

Select a flavor

Flavor is also selectable. The selected image or flavor can be used to set a default image or a default flavor.

```
flavor = mesh.flavor('india', 'm1.small')
```

Select an image

If you know the name of the vm image, you can specify the vm image to user it later. In this example, we choose Ubuntu trusty 14.04 image.

```
image=mesh.image('india','futuregrid/ubuntu-14.04')
```

Set a default flavor or image

```
defaults = mesh.default('india', 'image', image)

defaults = mesh.default('india', 'flavor', flavor)

pprint(defaults)
```

```
{u'_id': ObjectId('54242b0b6d8fecaafd838a11'),
 u'activeclouds': [u'india'],
 u'cm_user_id': u'gvonlasz',
 u'flavors': {u'india': u'2'},
 u'group': None,
 u'images': {u'india': u'ba327564-5969-4309-b3f3-b67764038e66'},
 u'index': u'41',
 u'key': u'flat-key',
 u'pagestatus': {u'india': u'true'},
 u'prefix': u'gvonlasz',
 u'project': u'fg82',
 u'registered_clouds': [u'india'],
 u'securitygroup': u'development',
 u'shell_print_format': u'table'}
```

```
image = "futuregrid/ubuntu-14.04"

flavor = "m1.small"

cloud = "india"
```

Quick Start a VM

A simple function *start* provides a quick launch of vm instances in cloudmesh.

```
result = mesh.start("india", username)

pprint (result)
```

```
{'cloud': 'india',
 'cm_user_id': 'gvonlasz',
 'flavor': u'm1.small',
 'flavor_id': u'2',
 'image': u'futuregrid/ubuntu-14.04',
 'image_id': u'ba327564-5969-4309-b3f3-b67764038e66',
 'key': u'gvonlasz_flat-key',
 'name': u'gvonlasz_41',
 u'server': {u'OS-DCF:diskConfig': u'MANUAL',
             u'adminPass': u'J6fnixrXGk5e',
             u'id': u'7547ecc9-61b8-42a4-9dac-0064a7407a63',
             u'links': [{u'href': u'http://149.165.146.57:8774/v1.1/8bc7e259464944b3bf4d8b050d1ab935/
                         u'rel': u'self'},
                        {u'href': u'http://149.165.146.57:8774/8bc7e259464944b3bf4d8b050d1ab935/serve
                         u'rel': u'bookmark'}],
             u'security_groups': [{u'name': u'default'}]}}
```

Delete VM

```
server = result['server']['id']

mesh.delete(cloud, server, username)
```

```
{'release_unused_public_ips': True, 'vm_delete': {'msg': 'success'}}
```

If you know the id of the virtual machine that you want to destroy, *delete* function in cloudmesh simply terminate the instance. This example deletes the vm that we just launched above by getting the id from the result dict.

Start a VM

Now let us see how to start VMs on a cloud, here is how to start a VM on cloud india.

```
result = mesh.start(cloud=cloud,
                    cm_user_id=username,
                    flavor=flavor,
                    image=image)
```

You may don't know what images or flavors are available on the cloud, or you don't want to type a long line every time you start a VM, things can get a lot easier by performing some setting up.

```
pprint (result)
```

```
{'cloud': 'india',
 'cm_user_id': 'gvonlasz',
 'flavor': u'm1.small',
 'flavor_id': u'2',
 'image': u'futuregrid/ubuntu-14.04',
 'image_id': u'ba327564-5969-4309-b3f3-b67764038e66',
 'key': u'gvonlasz_flat-key',
 'name': u'gvonlasz_42',
 u'server': {u'OS-DCF:diskConfig': u'MANUAL',
             u'adminPass': u'ETrauXf83o7M',
```

```
u'id': u'45b70984-12e1-40ec-a809-f46923ee3eac',
u'links': [{u'href': u'http://149.165.146.57:8774/v1.1/8bc7e259464944b3bf4d8b050d1ab935/
            u'rel': u'self'},
           {u'href': u'http://149.165.146.57:8774/8bc7e259464944b3bf4d8b050d1ab935/serve
            u'rel': u'bookmark'}],
u'security_groups': [{u'name': u'default'}]}}
```

More options to launch a VM instance

When you create a new VM instance, you can also choose multiple options such as a flavor, an image, or a key associated with the instance. *start()* function accepts keyword parameters as a user input of these options. To see a brief description of the function, try *mesh.start?* in the IPython Notebook cell.

Available options are: with * `cloud`: cloud id e.g. india * `cm_user_id`: portal user id * `prefix`: The VM instance name starts * `index`: The number of vm instances (auto increment), if you set this make sure it is higher than the last index, as it will aslo set the default index * `flavor`: flavor name e.g. m1.small * `image`: image name e.g. futuregrid/ubuntu-14.04 * `key`: key name to use * `meta`: data in python dict e.g. {"cm_user_id": username}

```
result = mesh.start("india", username, prefix=username)
```

Vitual Machine Name

In Cloudmesh, the default VM name consists of your username and an auto incremented number, for example, *alex_1*. Couple of functions allow you to manage or modify the VM name as you wish.

```
mesh.vmname()
```

```
u'gvonlasz_43'
```

`vmname()` without parameters returns the current VM name that you can use.

If you want to use a different name, you can specify prefix=" and idx=" as parameters. However be aware that using this will also reset the default values. Thus if you specify a number that is smaller than the current number in the vms, you may overwrite in time the previos name and end up with duplicated names.

```
mesh.vmname(username, 10)
```

vmname_next() returns a VM name with an increased index number.

```
mesh.vmname_next()
```

There are some tricks to update the index number in *vmname()* function.

```
mesh.vmname(username, "+5")
```

Assign a public IP address to the VM

assign_public_ip() function obtains a public IP address and assign it to the VM.

```
server = result['server']['id']
```

```
pprint (result['name'])
```

```
u'gvonlasz_42'
```

```
ip=mesh.assign_public_ip('india', server, username)
```

```
print ip
```

```
149.165.158.31
```

SSH to the VM

Once you obtained the public ip address, you can execute a test command via SSH to the VM. You can use wait() function with retry options. Otherwise, you can simply use ssh_execute() function.

```
result = mesh.wait(ipaddr=ip, interval=2, retry=10)
```

```
0 try to execute via ssh...
```

```
print result
```

```
True
```

```
mesh.ssh_execute(ipaddr=ip, command="uname -a")
```

```
Linux gvonlasz-42 3.13.0-35-generic #62-Ubuntu SMP Fri Aug 15 01:58:42 UTC 2014 x86_64 x86_64 x86_64
```

added previously added to your known host file. in that case you must remove it first from the ~/.ssh/known_hosts file.

Refreshing VM status

The information of VM instances can be displayed with `servers()` function.

```
mesh.refresh(username, names=["india"],types=["servers"])
```

```
* india
Refreshing gvonlasz servers india ->
Refresh time: 0.351860046387
Store time: 0.00319504737854
```

```
for serverid in mesh.servers(clouds=["india"],cm_user_id=username)["india"].keys():
    server =  mesh.servers(clouds=["india"],cm_user_id=username)["india"][serverid]
    print server['name']
```

```
gvonlasz_39
gvonlasz_42
gvonlasz_38
gvonlasz_40
gvonlasz-001
gvonlasz_38
```

7.4.6 Cloudmesh Mesh

Notebook

Warning: this example is not yet fully completed

```
import cloudmesh
```

```
username = cloudmesh.load().username()
```

One of the most convenient methods to interact in python is to use the cloudmesh mesh interface. To activate it use the mesh command. We are currently only supporting to run cloudmesh with the mongo db activated.

```
mesh = cloudmesh.mesh("mongo")
```

After activation we can find out more information with th einfo command.

```
mesh.activate(username)
```

```
mesh.refresh()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)

<ipython-input-9-6a9ebadaeb63> in <module>()
----> 1 mesh.refresh()


TypeError: refresh() takes at least 2 arguments (1 given)
```

7.5 Cloudmesh Shell

7.5.1 Cloudmesh Shell for VM Management

Notebook

Initialization

A simple import allows you to enable all features of Cloudmesh in Python.

```
import cloudmesh
```

```
print cloudmesh.shell("help")
```

```
Documented commands (type help <topic>):
========================================
EOF       dot2      help       list      plugins   script          var
banner    edit      image      loglevel  project   security_group  verbose
clear     exec      info       man       py        status          version
cloud     exp       init       metric    q         storm           vm
color     flavor    inventory  notebook  quit      timer           web
debug     graphviz  key        open      rain      use             yaml
defaults  group     label      pause     register  user

Ipython Commands
================
notebook

Gui Commands
============
web

Cloud Commands
```

```
==============
cloud     group  inventory  rain            status  vm     loglevel
defaults  image  list       register        storm   yaml   keys
flavor    init   metric     security_group  user    debug  project
```

```
print cloudmesh.shell("debug off")
```

```
Debug mode is off.
```

```
print cloudmesh.shell("loglevel error")
```

```
ERROR mode is set.
```

Activating Clouds

In order for cloudmesh to work with multiple coulds, we need to find out first which clouds are available. Users can add their own clouds later which we describe in the registration section.

Let us inspect what is already available by invoking the list command

```
print cloudmesh.shell("cloud list")
```

```
+-----------+--------+
| cloud     | active |
+-----------+--------+
| aws       |        |
| azure     |        |
| devstack  |        |
| dreamhost |        |
| hp        |        |
| hp_east   |        |
| india     |        |
+-----------+--------+
```

As you see we have a number of clouds, but none of them is already active. Thus we need to first activate a cloud. We assume that you have an account on FutureGrid. Let us activate the cloud india

```
print cloudmesh.shell("cloud on india")
```

```
* india
Refreshing gvonlasz servers india ->
Refresh time: 0.697345972061
Store time: 0.0153570175171
[32mcloud 'india' activated.[0m
```

We also have a conveniet interactive selector to select a cloud to work with, that however does not work with ipython

```
"cloud select"
```

or you may also input “cloud select india” to select a specific cloud india

To check if the cloud was activated, simply use the list command again

```
print cloudmesh.shell("cloud list")
```

```
+-----------+--------+
| cloud     | active |
+-----------+--------+
| aws       |        |
```

```
| azure     |         |
| devstack  |         |
| dreamhost |         |
| hp        |         |
| hp_east   |         |
| india     | True    |
+-----------+--------+
```

Start a VM

Now let us see how to start VMs on a cloud, here is how to start a VM on cloud india

```
print cloudmesh.shell("vm start --cloud=india --image=futuregrid/ubuntu-14.04 --flavor=m1.small")
```

```
* india
Refreshing gvonlasz servers india ->
Refresh time: 0.364186048508
Store time: 0.0172290802002
* india
Refreshing gvonlasz flavors india ->
Refresh time: 0.241857051849
Store time: 0.0082221031189
Refreshing gvonlasz images india ->
Refresh time: 0.592792987823
Store time: 0.0661771297455

# ########################################################################
# Starting vm->gvonlasz_14 on cloud->india using image->futuregrid/ubuntu-14.04, flavor->m1.small, ke
# ########################################################################
job status: PENDING
```

You may don't know what images or flavors are available on the cloud, or you don't want to type a long command every time you start a VM, things can get a lot easier by performing some setting up...

Set a default flavor or a default image

Each cloud must have a default image and a default flavor to launch vm instances in a simple step. The cloud set command provides a way to set default values for an image or a flavor.

```
print cloudmesh.shell("cloud set flavor india --flavorid=2")
```

```
* india
Refreshing gvonlasz servers india ->
Refresh time: 0.521083116531
Store time: 0.0117969512939
* india
Refreshing gvonlasz flavors india ->
Refresh time: 0.309951066971
Store time: 0.00973987579346
[32m'm1.small' is selected[0m
```

```
print cloudmesh.shell("cloud set image india --image=futuregrid/ubuntu-14.04")
```

```
* india
Refreshing gvonlasz servers india ->
Refresh time: 0.358379125595
```

```
Store time: 0.0185420513153
* india
Refreshing gvonlasz images india ->
Refresh time: 0.689934015274
Store time: 0.133810043335
[32m'futuregrid/ubuntu-14.04' is selected[0m
```

Get Flavors or Images

Available flavors can be listed with the following command.

```
print cloudmesh.shell("list flavor india --refresh")
```

refresh option updates the data from the IaaS cloud. The cached data in the mongo database will be updated.

The `list image` command provides an available vm images on a selected cloud.

```
print cloudmesh.shell("list image india --refresh")
```

Quick Start a VM

Once you configured a default flavor and a default image, you can start a new VM with a few options.

```
vmname = cloudmesh.shell("label --raw")
```

We use `vmname` to delete the vm instance later in this tutorial.

```
print cloudmesh.shell("vm start --cloud=india --name={0}".format(vmname))
```

Delete a VM

If you know the id or name of the virtual machine that you want to destroy, *delete* command in cloudmesh simply terminate the instance.

```
cloudmesh.shell("vm delete {0} --cloud=india --force".format(vmname))
```

> **Warning:** Do not execute the delete command as it deletes all vms of

the tennant in openstack.

More options to launch a VM instance

When you create a new VM instance, you can also choose multiple options such as a flavor, an image associated with the instance. The `vm start` command accepts optional parameters as a user input of these options. To see a brief description of the command, try `cm "vm --help"` in the IPython Notebook cell.

Available options are:

- –cloud= give a cloud to work on, if not given, selected or default cloud will be used
- –count= give the number of servers to start
- –flavor= give the name of the flavor
- –flavorid= give the id of the flavor

- –group= give the group name of server
- –image= give the name of the image
- –imageid= give the id of the image

```
cloudmesh.shell("vm start --cloud=india --flavor=m1.medium --image=futuregrid/ubuntu-12.04")
```

Set a default cloud

If you want to make things even more convenient, you can set a default cloud or select a cloud to work with so that you don't have to type in a cloud everytime you need to specify a cloud, to set india as default cloud

```
print cloudmesh.shell("cloud set default india")
```

to select a cloud

```
print cloudmesh.shell("cloud select india")
```

You can see a selected cloud as a temporarily default cloud to work with.

For more details of using command cloud to set up a cloud

```
print cloudmesh.shell("cloud -h")
```

After all setting up above, now you can start a VM simply by typing in

Refreshing VM status

After you have started or deleted VMs, you may want to check clouds' VMs status. To refresh cloud india's VMs information

```
print cloudmesh.shell("list vm india --refresh")
```

Starting multiple VMs

Sometimes we want to start more than one VM at the same time, we can choose the option –count=int where int is the number of VMs you want to start. For example, to start 3 VMs on india

```
import uuid;
temp_group_name="ipython-tutorial-" + str(uuid.uuid4().get_hex().upper()[0:6])
```

We use the temporary group name to delete the VMs later in this tutorial.

```
print cloudmesh.shell("vm start --cloud=india --count=3 --group={0}".format(temp_group_name))
```

Deleting VMs

To delete one VM is easy, what if we want to delete 1000 VMs, we need a more convenient way to do it. Cloudmesh shell provides several methods to find the VMs and delete them, you may think there are two phases of VM deletion, searching and deleting. Here are some examples:

Let's start with deleting the VMs that we started in the previous example. We can delete them with the group name that we used. `temp_group_name`

```
cloudmesh.shell("vm delete --group={0} --cloud=india --force".format(temp_group_name))
```

to delete all VMs of cloud india

```
cloudmesh.shell("vm delete --cloud=india --force")
```

Note here we use the option "–force", without it the shell will give you a list of VMs to delete and ask for your confirmation.

to delete a VM by giving its name (you may always provide a cloud unless you have specified a default cloud or have selected a cloud)

```
cloudmesh.shell("vm delete --cloud=india sample_vm --force")
```

to delete a VM by group

```
cloudmesh.shell("vm delete --cloud=india --group=testgroup --force")
```

We can also narrow the search result by giving more search conditions. For example, to delete VMs of cloud india that they are also in the group 'testgroup' and they have the prefix name 'abc' and their indices' range is no greater than 100

```
cloudmesh.shell("vm delete --cloud=india --group=testgroup --prefix=abc
--range=,100 --force")
```

Login to a VM

To login to a VM, you need to assign a public id to the VM you started, for example, assign a public ip to a VM named test_1 on cloud india:

```
vmname = cloudmesh.shell("label --raw")

print cloudmesh.shell("vm start --cloud=india")

cloudmesh.shell("vm ip {0} --cloud=india".format(vmname))
```

Then you can login to this VM by(note here you need to provide the login name for your VM, it varys depend on the image you use while you start the VM, e.g. for ubuntu, you may type –ln=ubuntu):

cloudmesh.shell("vm login {0} –ln=ubuntu –cloud=india".format(vmname))

IPython Notebook does not support interactive shell for ssh login. The command above can be executed in the Python console.

If you just want to run some commands and get the return on the VM, you may add the commands at the end following '–'(e.g. ls -a)

```
cloudmesh.shell("vm login {0} --ln=ubuntu --cloud=india -- ls -a".format(vmname))
```

For more details for command vm

print cloudmesh.shell("vm -h")

```
cloudmesh.shell("vm delete {0} --cloud=india".format(vmname))
```

Stop the vm that we started for this tutorial.

Vitual Machine Name

In Cloudmesh, the default name of VM consists of your username and a number, for example, alex_1. `label` command allows you to manage or modify the VM name as you wish.

```
cloudmesh.shell("label")

next vm name:
albert_71
```

cloudmesh.shell("label –prefix=albert –id=40")

If the user doesn't provide a name while starting VMs, cloudmesh will generate labels for them. The default form to name VMs is prefix_index, where prefix is a string and index is an non-negative integer. If a index is used, the index value will be automatically added by one waiting to be used for next VM. To check your current prefix and index

7.5.2 Cloudmesh Shell - list command

The *list* command provides cloud information about flavors, images, vm instances, projects, IaaS clouds available as part of your federated cloudmesh infrastructure. Some examples in this document lets you understand what information can be viewed in the cm command line shell.

Notebook

```
import cloudmesh

cloudmesh.shell('help list')

List available flavors, images, vms, projects and clouds

    Usage:
        list flavor [CLOUD|--all] [--refresh] [--format=FORMAT]
        [--column=COLUMN]
        list image [CLOUD|--all] [--refresh] [--format=FORMAT] [--column=COLUMN]
        list vm [CLOUD|--all] [--refresh] [--format=FORMAT] [--column=COLUMN]
        list project
        list cloud [--column=COLUMN]

    Arguments:

        CLOUD    the name of the cloud e.g. alamo, india

    Options:

        -v         verbose mode
        --all      list information of all active clouds
        --refresh  refresh data before list

        --column=COLUMN        specify what information to display in
                               the columns of the list command. For
                               example, --column=active,label prints
                               the columns active and label. Available
                               columns are active, label, host,
                               type/version, type, heading, user,
                               credentials, defaults (all to display
                               all, email to display all except
                               credentials and defaults)
```

```
        --format=FORMAT         output format: table, json, csv

    Description:

        List clouds and projects information, if the CLOUD argument is not specified, the
        selected default cloud will be used. You can interactively set the default cloud with the com
        'cloud select'.

        list flavor
        : list the flavors
        list image
        : list the images
        list vm
        : list the vms
        list project
        : list the projects
        list cloud
        : same as cloud list

    See Also:

        man cloud
```

Listing flavors

You can see available flavors with *list flavor* command.

```
print cloudmesh.shell("list flavor india")
```

```
+------+-----------+---------+-------+--------+----------------------+
|   id | name      |   vcpus |   ram |   disk | refresh time         |
+======+===========+=========+=======+========+======================+
|    1 | m1.tiny   |       1 |   512 |      0 | 2014-09-05T09-28-34Z |
+------+-----------+---------+-------+--------+----------------------+
|    3 | m1.medium |       2 |  4096 |     40 | 2014-09-05T09-28-34Z |
+------+-----------+---------+-------+--------+----------------------+
|    2 | m1.small  |       1 |  2048 |     20 | 2014-09-05T09-28-34Z |
+------+-----------+---------+-------+--------+----------------------+
|    5 | m1.xlarge |       8 | 16384 |    160 | 2014-09-05T09-28-34Z |
+------+-----------+---------+-------+--------+----------------------+
|    4 | m1.large  |       4 |  8192 |     80 | 2014-09-05T09-28-34Z |
+------+-----------+---------+-------+--------+----------------------+
```

As you can see, you may need to specify cloud name to see, for example *sierra_openstack_grizzly* in this example.

Listing images

You can see available images with *list image* command.

```
print cloudmesh.shell("list image india --column=name,id")
```

```
+-------------------------------------------------+--------------------------------------+
| name                                            | id                                   |
+=================================================+======================================+
| salsahpc/cloud-mooc-m1-large-4GB                | 384ca88c-f674-4d3d-999f-353fc6915608 |
+-------------------------------------------------+--------------------------------------+
```

```
| CentOS6                                                | ad35042c-d242-4514-bde0-138718b5aa3e |
+--------------------------------------------------------+--------------------------------------+
| SL64-blank-sparse10gb-C vmdk                           | f480f1e7-38af-4410-bc13-6a1e43bf58ba |
+--------------------------------------------------------+--------------------------------------+
| fg7/rynge/centos6-v1                                   | ac464f65-7175-44df-9e0e-34b1a32f8a2a |
+--------------------------------------------------------+--------------------------------------+
| futuregrid/centos-6                                    | 81b27cb5-4f8b-4583-afd4-1901053f6a28 |
+--------------------------------------------------------+--------------------------------------+
| grp17Cent                                              | 650a3716-e4b9-4da7-a2f7-7e9a67a971b0 |
+--------------------------------------------------------+--------------------------------------+
| DaLiAna-vm2014e-geant4.10.vmdk Jan best                | f4f9821f-881b-433a-97ab-1ad87273c5fc |
+--------------------------------------------------------+--------------------------------------+
| ubuntu-13.10                                           | cd844b12-f138-414a-8cd5-2c977f0a2379 |
+--------------------------------------------------------+--------------------------------------+
| futuregrid/ubuntu-14.04                                | ba327564-5969-4309-b3f3-b67764038e66 |
+--------------------------------------------------------+--------------------------------------+
| fg101/richieriee/my-ubuntu-01                          | 7915f335-d1a9-4380-a9f3-159eb67d721e |
+--------------------------------------------------------+--------------------------------------+
| fg419/hsiychen/ubuntu-master                           | 0f6aa6b4-0594-4d79-855e-94f748add164 |
+--------------------------------------------------------+--------------------------------------+
| sl64-gluex-vm2014e-40gb.vmdk Justin                    | f51fc1e4-d495-4376-824b-19000c41abef |
+--------------------------------------------------------+--------------------------------------+
| futuregrid/fedora-19                                   | 6e9322df-fcb2-4ac8-bd02-1d39c6f6919f |
+--------------------------------------------------------+--------------------------------------+
| fg10/jcharcal/centos6.5_x86_64                         | 28f2ad75-8f42-4079-85e1-3d4fe986317f |
+--------------------------------------------------------+--------------------------------------+
| balewski/kernel-2.6.32-431.5.1-sl65                    | 00e935c3-82a6-499c-9056-6db37d27439b |
+--------------------------------------------------------+--------------------------------------+
| SL64-blank-sparse40GB vmdk                             | 882871a5-3157-426a-b7bc-10cca91860f0 |
+--------------------------------------------------------+--------------------------------------+
| Ubuntu-12.04-blank2.vmdk                               | b49324f8-d2c3-413a-8f8c-83b498e815f6 |
+--------------------------------------------------------+--------------------------------------+
| balewski/ramdisk-2.6.32-431.5.1-sl65                   | 9589cf93-f0e6-45d8-930f-63dd2d9c2f1a |
+--------------------------------------------------------+--------------------------------------+
| sl6_x64-qemu  french ?bad                              | c9ca9852-dfcb-4ceb-a164-a58c89594551 |
+--------------------------------------------------------+--------------------------------------+
| balewski/sl6.5-blank-80gb-b works                      | 5cb90f55-c3a1-468b-b60a-0ecd589baf31 |
+--------------------------------------------------------+--------------------------------------+
| futuregrid/ubuntu-12.04                                | 9cd8cc0d-96cd-44b8-a307-8b574c899ef4 |
+--------------------------------------------------------+--------------------------------------+
| ndssl-vt/ubuntu-12.04-small                            | b9455041-407b-488b-a3f1-5dce6c480b5a |
+--------------------------------------------------------+--------------------------------------+
| futuregrid/fedora-20                                   | 68a5d085-ea4d-42c4-800f-2017309989ae |
+--------------------------------------------------------+--------------------------------------+
| cglmoocs/ipython                                       | 0eff8bfc-e455-429e-9dfb-e16488d410f9 |
+--------------------------------------------------------+--------------------------------------+
| balewski/daliana-vm2014e2-sl6.5-geant4.10-root5.34     | 64866d96-8a9e-4d7c-983f-90b0bb594310 |
+--------------------------------------------------------+--------------------------------------+
| DaLiAna-vm2014d-SL64-A.vmdk                            | ff5bb4de-d7e9-4bfb-b42b-cec9ed8d2d08 |
+--------------------------------------------------------+--------------------------------------+
```

With the –column option, you can select particular value(s) to view. In this example, we choose name, id, updated columns to view.

- Note that *–column=[column names separated by comma (,)]* displays one or more columns selected or *–column=all* displays all columns.

Listing vm instances

list vm command provides the status of scheduled/launched/terminated virtual machines.

```
print cloudmesh.shell("list vm india --column=status,name,address,flavor,id,created")
```

```
+--------------+----------+--------------------------------------+----------+----------------------+
| name         | status   | id                                   | flavor   | created              |
+==============+==========+======================================+==========+======================+
| gvonlasz-002 | ACTIVE   | 211563da-7d18-41f1-85b7-eb959dbe1c75 | m1.small | 2014-09-12T19:50:55Z |
+--------------+----------+--------------------------------------+----------+----------------------+
```

JSON format output

With *–format=json* option, the result is displayed in json.

```
print cloudmesh.shell("list vm india --format=json")
```

If you use *–format* option, the result will be displayed in the selected format. At this time, *–format* supports 'json' and 'table' formats. csv or html might be added later.

Options

Some options provide different output of *list* command.

- –all: list information for all clouds. [CLOUD] name is not required.
- –refresh: update the information from the server. Cached data in the mongo db will be updated.
- –column=COLUMN: column name(s) separated by comma (,) to view the selected information. –column=all displays all.
- –format=FORMAT: output format, table, json is available.

7.6 Cloudmesh GUI

7.6.1 Screenshots

A shot video about the Cloudmesh GUI is available at 

l_P4G85rysA.

Please note that cloudmesh supports a role based user policy model. Although you may see some screenshots of advanced features some of these features may not yet released to the users.

Cloud Management

Cloudmesh has a simple interface to register and conduct some elementary management functions. In contrast to other systems cloudmesh uses the native cloud protocol and is not just relying on the EC2 compatible clouds. Certainly the Graphical user interface can be improved and customized. We have just provided a very simple interface that focuses on exposing more info that encourages you to conduct more with the management functionality instead of just hiding information to the user. For end-users, we can naturally develop a much simpler interface, as for example is demonstrated in our launcher (which is not yet released).

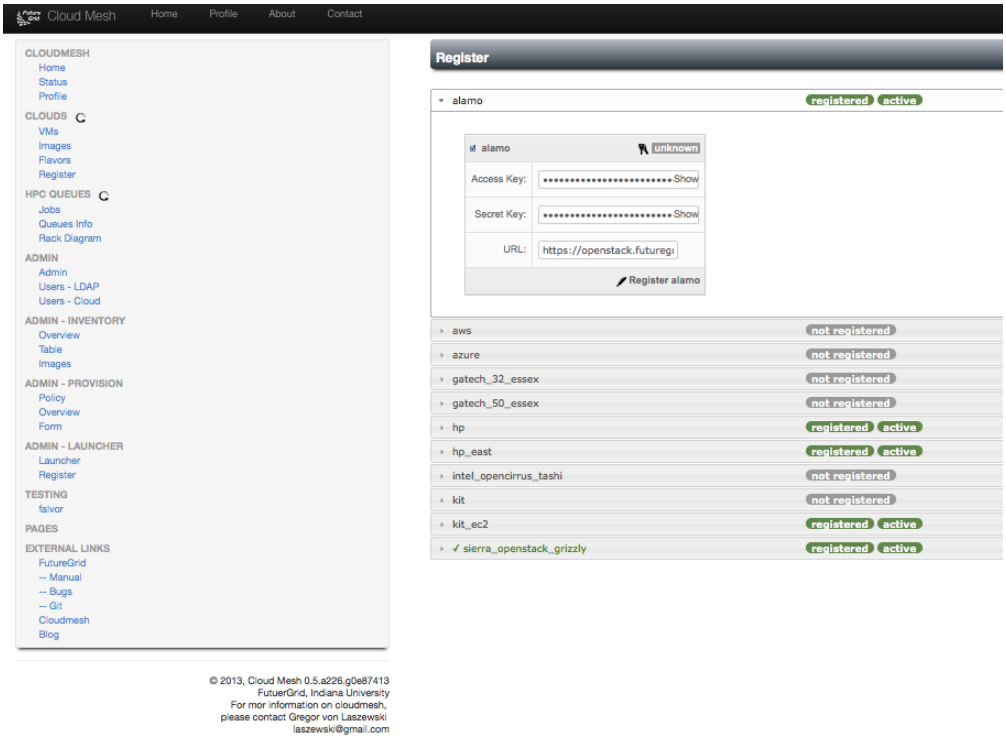


Figure 7.4: **Figure:** Registering an EC2 compatible cloud

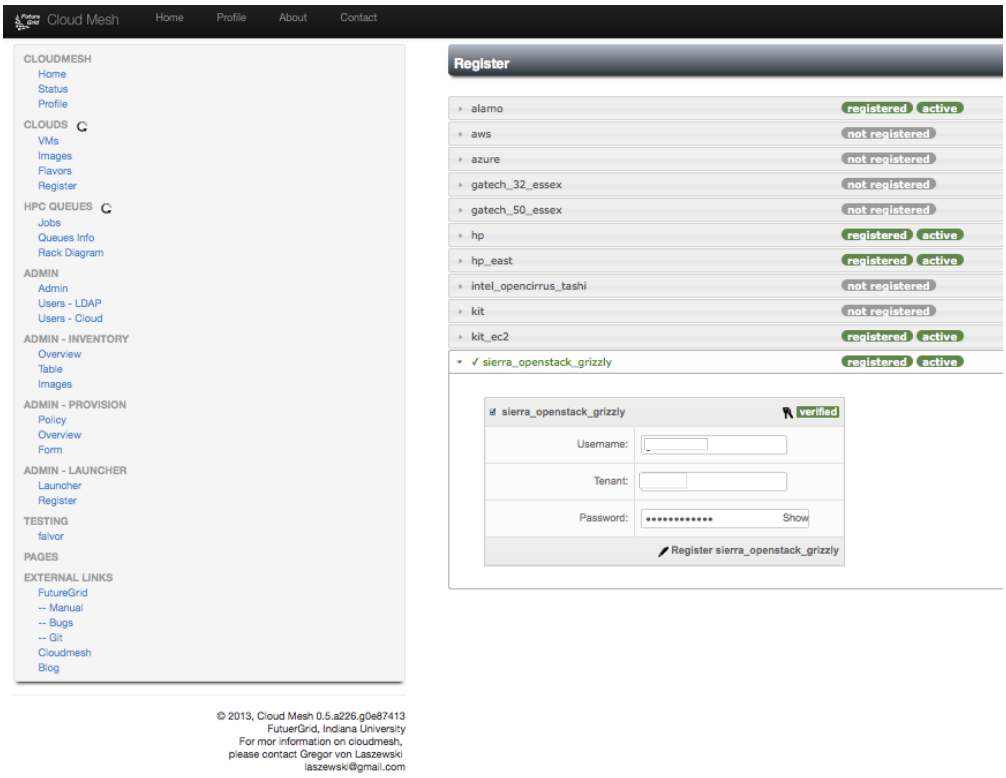


Figure 7.5: **Figure:** Registering an OpenStack protocol compatible cloud.

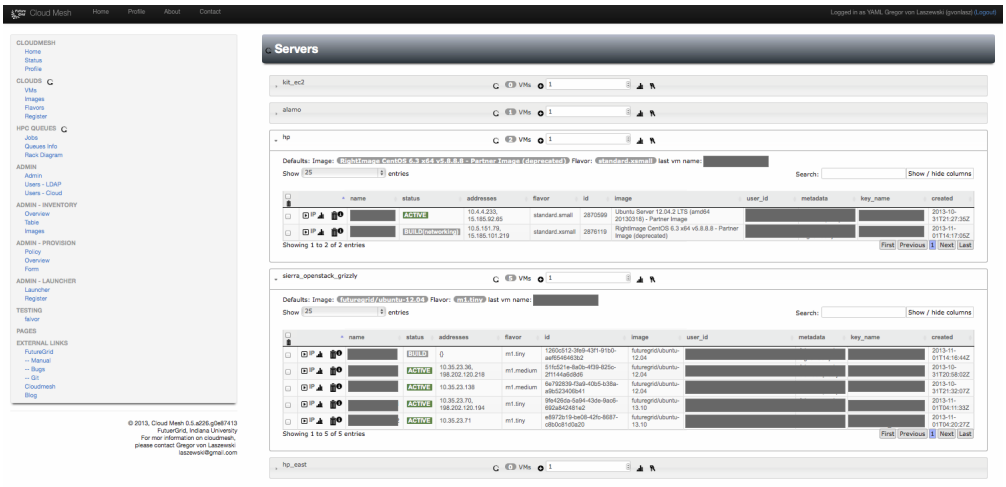


Figure 7.6: **Figure:** Starting and deletion of VMs is easy in cloudmesh through a simple table view.

Provisioning/Raining (Not yet released)

Cloudmesh contains the ability to provision a server via bare metal access by the users. To simplify this already available access we are currently developing a simpler interface to it. We have already implemented a policy based access control that allows a role based access based on projects and users. In near future we will integrate our bare metal provisioning management. features into this system.

Batch Queues

Hadoop is often installed on a cluster. Thus having access to the queues to monitor queue based resource reservation for Hadoop jobs (or and other HPC job) is conveniently provided in cloudmesh. Launchers (under development) can be used to easily interface with the systems and conduct customized job creation. Via MyHadoop for example it is possible to start Hadoop jobs in queues on FutureGrid.

Status

The status of the system will be visible in a status window. Here we just show a view of the HPC resources. We already have developed a cloud monitoring system that we intend to integrate soon. For FutureGrid this system is already deployed via the FG portal.

Inventory

Often we just need to know sme details about the system. To facilitate this, we have developed an inventory. In addition we also developed physical view of the rack that can either be augmented with service type displays or temperature of the rack.

7.6.2 Enabeling ssh in the GUI

The current terminal is only supported if the server runs on the machine that executes the ssh terminal.

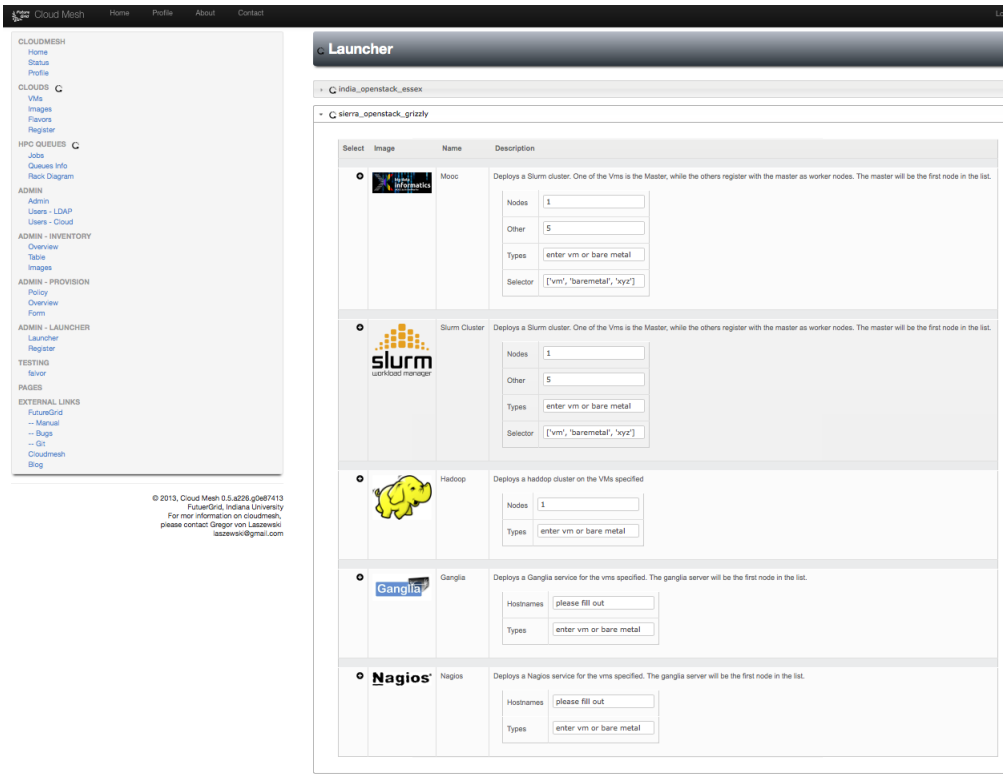


Figure 7.7: **Figure:** Launching predefined configurations on FutureGrid

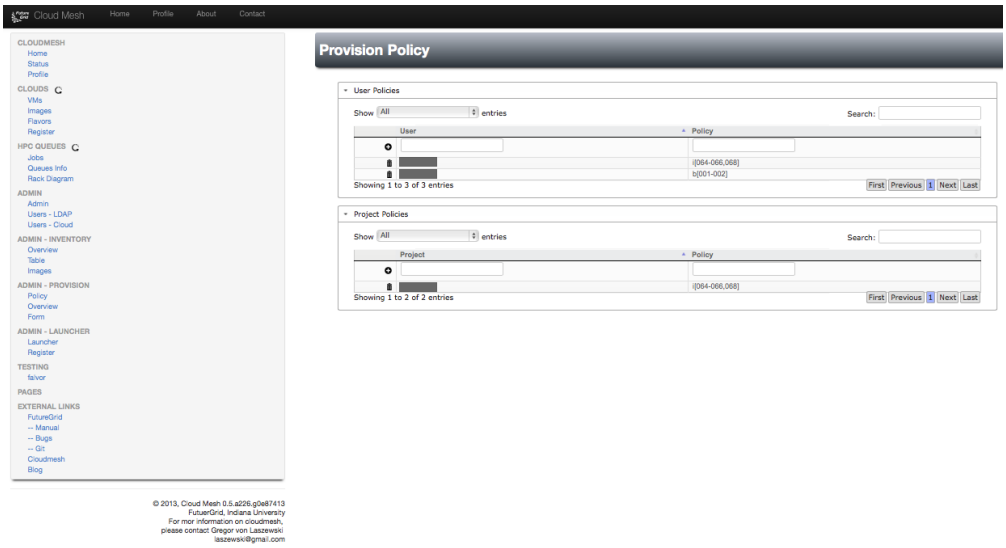


Figure 7.8: **Figure:** Defining the baremetal access policy

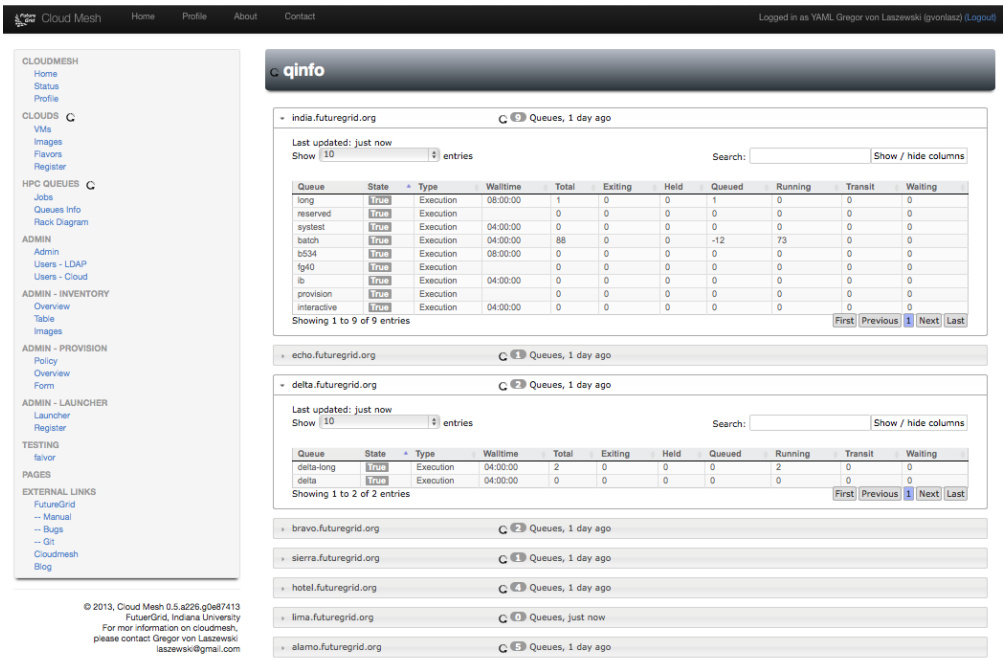


Figure 7.9: **Figure:** Listing the available queues

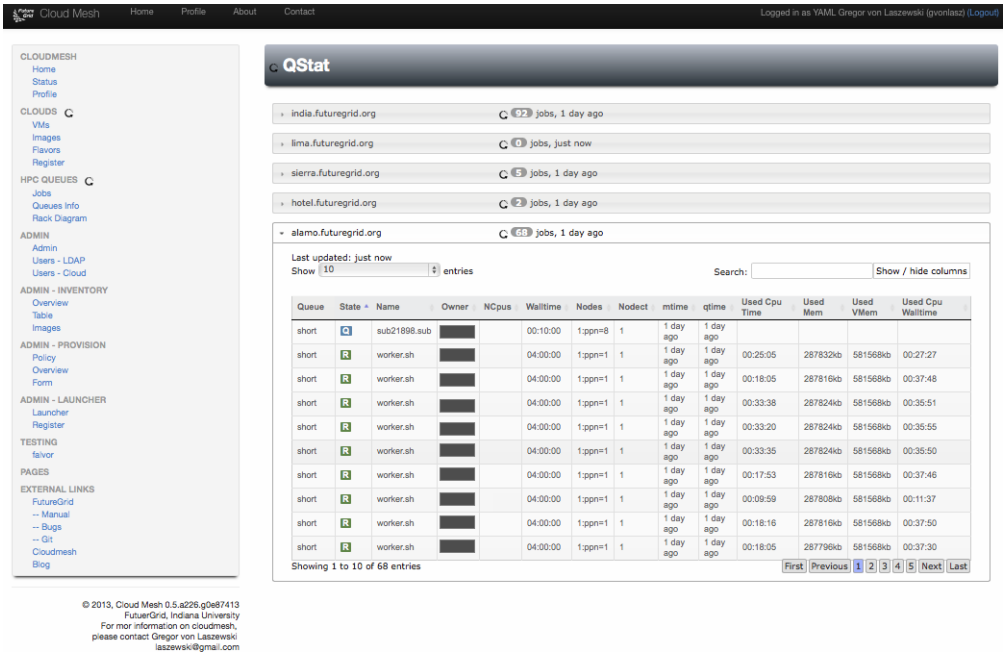


Figure 7.10: **Figure:** Listing the queue information about jobs and status

Figure 7.11: **Figure:** Displaying a simple status of the systems (here HPC).

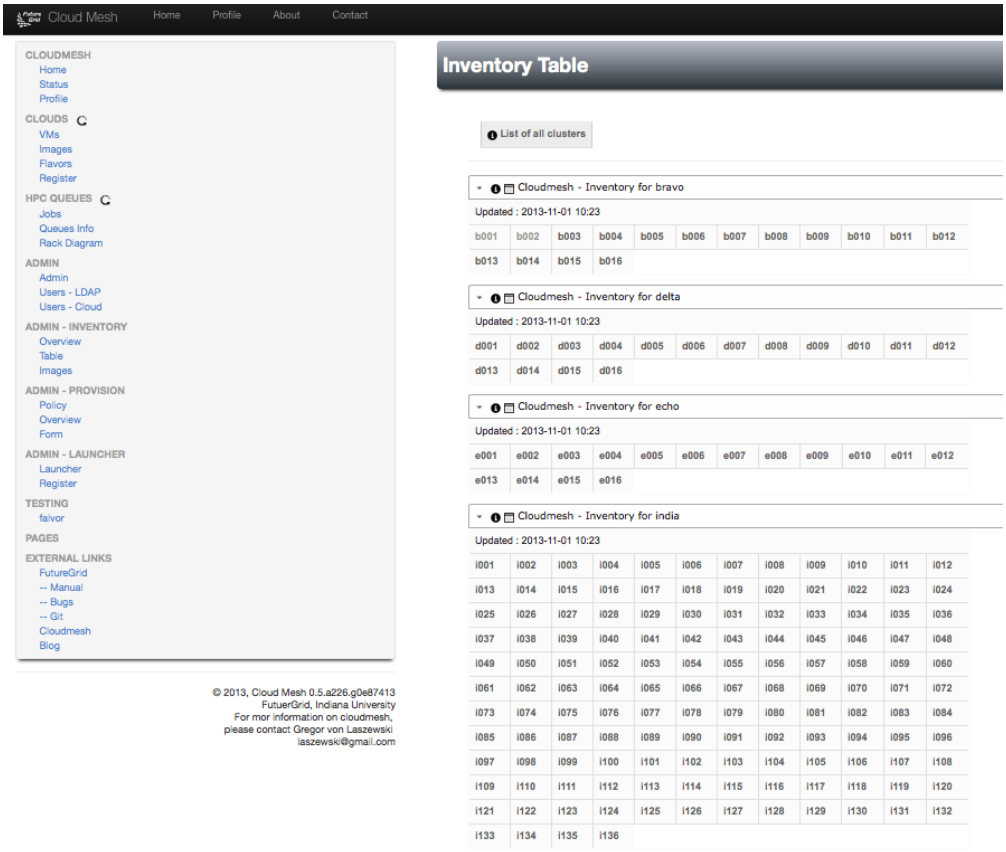


Figure 7.12: **Figure:** Inventory of the systems.

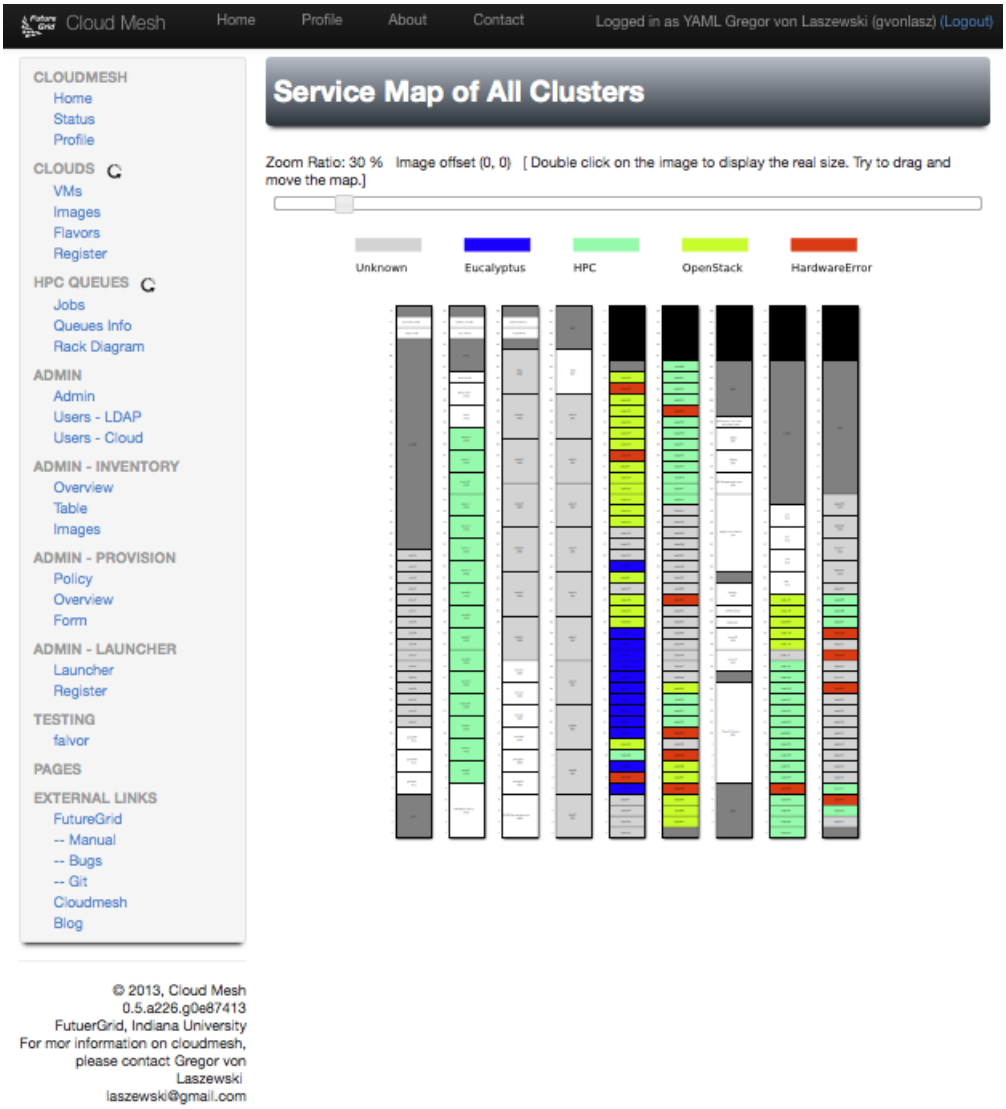


Figure 7.13: **Figure:** Service map to depict which server is dedicated to which services

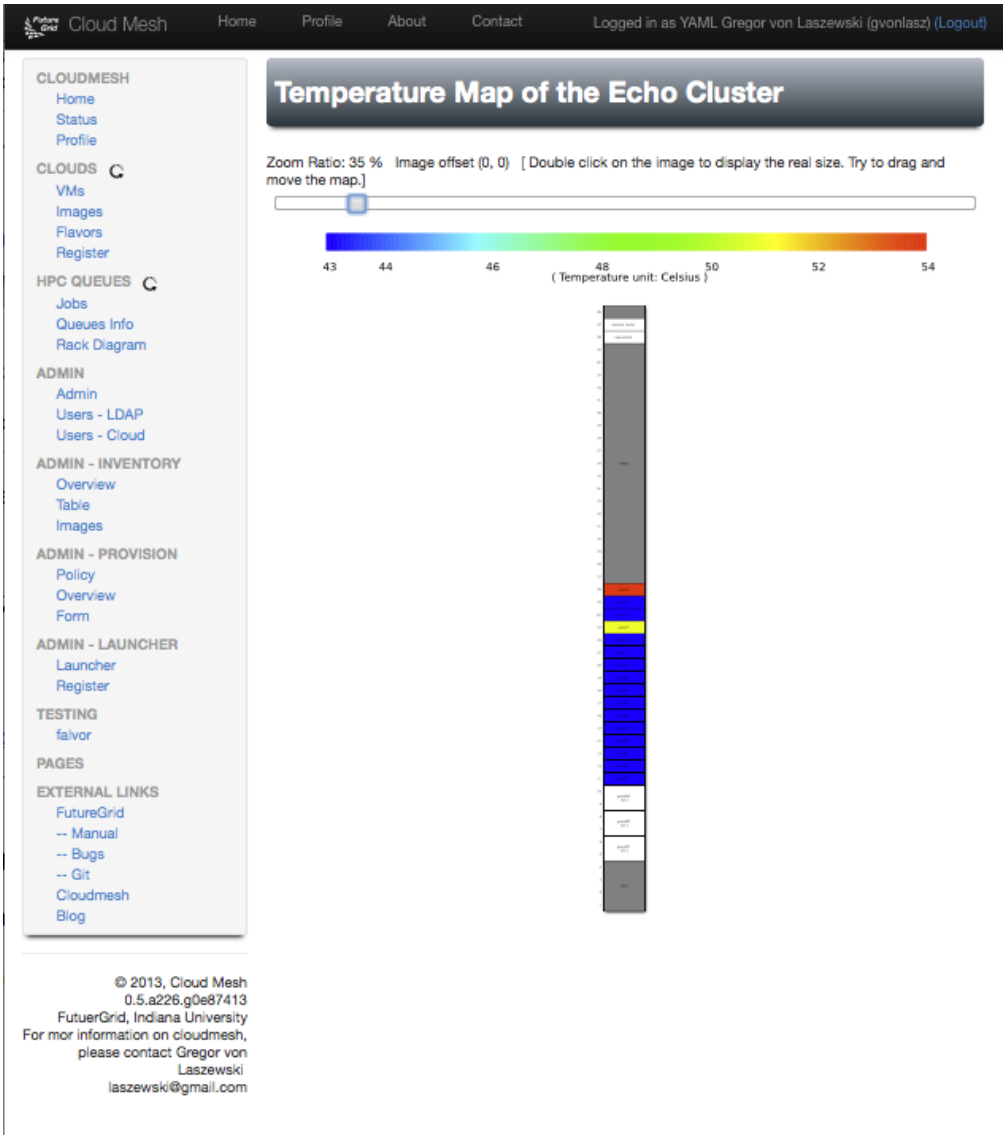


Figure 7.14: **Figure:** Temperature map of a rack

Safari

Safari comes with support for the *ssh://* protocol and is thus immediatly available if the server is started on the local machine.

Firefox

In case you like to enable the ssh protocol in firefox, please install FireSSH.

Chrome

Chrome is at this time not supported as the crome ssh plugin has a different syntax. We have not enabled yet access to it.

7.7 Cloudmesh FAQ

7.7.1 My password does not work

Remember that the password for cloudmesh is not your Opemstack password or is it your portal password. THis is done to be more secure and separate thes epasswords. The password is created during the setup process. If you forgot this password wuo can reset it with:

```
cd cloudmesh
fab user.mongo
```

CHAPTER

EIGHT

PARALLEL SHELL

Traditionally system administrators and developers using parallel computing need tools to manage a significant number of machines. One of the requirements is to execute a command in parallel on many machines and gather its output. There are many tools that can achieve this task. We focus here on the introduction of the following tools:

1. pdsh - a parallel distributed shell
2. fabric - python framework to execute commands on remote computers
3. Cloudmesh Sequential and Parallel python functions for executing repeatedly commands with caching

8.1 Parallel Distributed Shell (pdsh)

The parallel distributed shell (pdsh) is a shell command line program that allows the execution of commands not just on one computer but on a list of computers.

An online version of the manual pages is located at

- http://linux.die.net/man/1/pdsh

An important feature is that the list of hosts can be specified in a convenient form that is also kwon as hostlists. This format allows you to define a list of hosts based on some abbreviation. For example the string:

```
host[0-3]
```

will create a host list containing the hosts:

```
host0, host1, host2, host3
```

Furthermore, substitutions for the user and the hostname to login to the remote machine while leveraging ssh config files make this tool real easy to use. One such example:

```
pdsh -R exec -w host[0-3] ssh -x -l %u %h hostname
```

executes the command hostname on all specified machines.

Todo

Hyungro, check with allan and Koji if we have pdsh on india. At this time we are not aware that pdsh is installed by default on india. check with the systems group and have them provide a documentation on how we activate it.

8.2 Fabric

Fabric is a Python command-line tool and library for assisting system administration tasks related to the execution of command via ssh. It includes the ability to execute commands on the local machine, but also on a remote machine. Due to the integration with Python function definitions, it has also somewhat the ability to write "target" like specifications that we may know from makefiles. However it is more sophisticated as we can use the full feature richness of python.

The web page of Fabric which includes several examples and tutorials is located at:

- http://www.fabfile.org

Similar to the previous command we like to start the command hostname on a number of machines. To install fabric you simply say:

```
pip install fabric
```

in your virtualenv. Next, we define a file called fabfile.py with the following contents:

```
from fabric.api import run

def hostname():
    run('hostname')
```

Next let us run this command on the local computer and test it out:

```
fab -H localhost hostname
```

This will execute the function defined with the name hostname and print it via the run command. To execute the command on multiple hosts, you can simply specify them as part of the -H argument:

```
fab -H host0,host1,host2,host3 hostname
```

8.3 Cloudmesh Parallel API

The previous commands are all developed with a single user in mind, i.e. a single user executes the command. However in the age of cloud computing what would happen if thousands of users were to execute the same task, or even when ten users execute the same task, but the task would take considerable compute time to calculate? The answer is obvious, we would waste valuable compute resources as we do not take into consideration that the same task may be run by multiple people. To overcome this challenge we have started a simple demonstration program in our cloudmesh repository to partially address this issue.

To do so we are at this time we are only focussing on the consecutive execution of a command in a particular time period. Instead of executing the command over and over, we will simply return the result from a result cache. The cache has a specific time to life in which no new results are created but the result is read from the cache. New requests are cached.

We recognize that the example we provide is not a complete solution to our problem, but a step in the right direction. I also has the advantage of being relatively simple and introducing you to a number of tools and concepts that will become important when dealing with parallelism in the cloud.

8.3.1 Requirements

1. A computer with python 2.7
2. Using a python virtualenv
3. Having downloaded the cloudmesh code with git clone as discussed elsewhere

4. Having installed the cloudmesh code and libraries as discussed elsewhere

8.3.2 Code

The code is located in the directory:

```
cloudmesh_examples/example_mongo
```

within the cloudmesh code you have cloned from github. The code contains two python functions called Sequential and Parallel, that allow users to run commands either sequentially or in parallel on a number of hosts. The hosts can be specified in a yaml file located in:

```
~/.cloudmesh/cloudmesh_hpc.yaml
```

An example would be:

```
meta:
  yaml_version: 3.0
  kind: hpc
cloudmesh:
    hpc:
        alamo:
            cm_host: alamo.futuregrid.org
            cm_type: hpc
            username: albert
        india:
            cm_host: india.futuregrid.org
            cm_type: hpc
            username: albert
        sierra:
            cm_host: sierra.futuregrid.org
            cm_type: hpc
            username: albert
        bigred:
            cm_host: bigred2.uits.iu.edu
            cm_type: hpc
            username: albert
```

This file is used to specify the username for each host and define the host names. In case you want to run commands on the hosts you can do this with the following python program.

The first command executes the task sequentially over the array given in the first parameter. The second one executes it in parallel. Instead of just presenting you with a bare bones program we present you with some additional features that are worth noting and may come in handy in future. This includes the availability of a named stopwatch and the ability to read configuration parameters easily from a yaml file. Sometimes it is also nice to have very visible debug messages that we create with a banner function. Results are often more readable when using the python pprint function instead of just the print function. This is especially true when we print data-structures such as arrays and dicts. Next we will present the program and explain a selected number of features by commenting them in the code. We assume you know by now elementary python.

```
from cloudmesh_task.tasks import cm_ssh
from cloudmesh_task.parallel import Parallel, Sequential
from cloudmesh.util.stopwatch import StopWatch
from cloudmesh_common.util import banner
from pprint import pprint
from cloudmesh.config.cm_config import cm_config
from cloudmesh.config.ConfigDict import ConfigDict
import sys
```

```
# read the information from the yaml file into a dict called config
config = ConfigDict(filename="~/.cloudmesh/cloudmesh_hpc.yaml")["cloudmesh"]["hpc"]

# a function to extract from the config file the username from all
# hostnames in the array hosts
def  get_credentials(hosts):
    credential = {}
    for host in hosts:
        credential[host] = config[host]['username']
    return credential

# find all hostnames from the config file
hosts = config.keys()

# find all credentials (username, hostname) from the hosts in the
#  config file
credentials = get_credentials(hosts)


# create a stop watch
watch = StopWatch()

# execute is a python function. It is either Parallel or Sequential
# * modify
#    for execute in [Sequential]:
#    for execute in [Parallel]:

for execute in [Sequential, Parallel]:

    # get the name of the function
    name = execute.__name__

    # print the name of the function and start the timer
    banner(name)
    watch.start(name)

    # execute the function and return the result in a dict
    result = execute(credentials, cm_ssh, command="qstat")

    # stop the timer and print the result dict
    watch.stop(name)
    pprint(result)

    # only print the output from the command we executed
    banner("PRINT")
    for host in result:
        print result[host]["output"]

# print the timers
for timer in watch.keys():
    print timer, watch.get(timer), "s"
```

Bug: Before you start the command, you have to start a new window and say fab fab manage.mongo in the cloudmesh directory where your fabfiles are located. This will give something like:

```
$ fab manage.mongo
[localhost] local: make -f cloudmesh/management/Makefile mongo
```

```
mongod --noauth --dbpath . --port 27777
all output going to: /usr/local/var/log/mongodb/mongo.log
```

To run the command you will need to start the caching backend services. to do so we created a simple program cm-task.py that will be used to start and stop the services:

```
./cm-tasks.py menu

Queue Management
================

    1 - all start
    2 - all stop
    3 - rabbit start
    4 - celery start
    5 - rabbit stop
    6 - celery stop
    7 - mongo start
    q - quit

Select between 1 - 7:
```

Now select the number:

```
1 - all start
```

This will bring up the necessary services and look similar to:

```
 -------------- celery@host.local v3.1.13 (Cipater)
---- **** -----
--- * ***  * -- Darwin-13.3.0-x86_64-i386-64bit
-- * - **** ---
- ** ---------- [config]
- ** ---------- .> app:         cloudmesh_task:0x10365bcd0
- ** ---------- .> transport:   amqp://guest:**@localhost:5672//
- ** ---------- .> results:     amqp
- *** --- * --- .> concurrency: 10 (prefork)
-- ******* ----
--- ***** ----- [queues]
 -------------- .> celery           exchange=celery(direct) key=celery


[tasks]
  . cloudmesh_task.tasks.cm_ssh

[2014-08-19 15:46:24,060: INFO/MainProcess] Connected to amqp://guest:**@localhost:5672//
[2014-08-19 15:46:24,071: INFO/MainProcess] mingle: searching for neighbors
[2014-08-19 15:46:25,098: INFO/MainProcess] mingle: sync with 10 nodes
[2014-08-19 15:46:25,099: INFO/MainProcess] mingle: sync complete
[2014-08-19 15:46:25,109: WARNING/MainProcess] celery@host.local ready.
[2014-08-19 15:46:28,352: INFO/MainProcess] Events of group {task} enabled by remote.
```

After this you can start the program repeatedly with:

```
$ python prg.py
```

We are committing some of the output but at the end ist should look something like:

```
# ########################################################################
# PRINT
```

```
# ###################################################################
Tue Aug 19 15:48:29 EDT 2014
Job id                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
1589570.i136               sub18248.sub     aaaa                   0 Q delta
1589589.i136               sub15366.sub     aaaa                   0 Q delta
1589669.i136               sub12428.sub     aaaa                   0 Q delta
1795838.i136               twisterJob       bbbbbb                 0 Q batch
1872981.i136               sub9593.sub      aaaa                   0 Q delta
1904453.i136               sub2114.sub      aaaa                   0 Q delta
1904930.i136               dimer_in_sol_ph7 cccccccc        883:55:5 R batch
1904931.i136               dimer_in_sol_ph5 cccccccc        902:18:4 R batch
1904957.i136               suffix           dddddddd        360:36:1 R echo
1904961.i136               dimer_in_sol_ph7 cccccccc               0 H batch
1904963.i136               dimer_in_sol_ph5 cccccccc               0 H batch
1904993.i136               blast            eeee            15:08:00 R bravo
1904995.i136               blast            eeee            14:33:19 R bravo
1905016.i136               papi-inca        aaaa                   0 Q bravo
1905021.i136               vampir-inca      aaaa                   0 Q bravo
1905044.i136               papi-inca        aaaa                   0 Q bravo
1905057.i136               STDIN            ffffffff        00:10:17 R delta
1905062.i136               ...Script.i21500 gggggg          00:00:11 R batch

Sequential 12.12866169 s
Parallel    0.00446796417236 s
```

Please note that we have replaced the real usernames.

When you execute this command you will notice That the parallel execution time is much faster. In this case it was within the TTL and thus read the cache value from the cache. Executing the command again within the TTL will give you also for the sequential time a real short value:

```
Sequential 0.00726103782654 s
Parallel   0.000990867614746 s
```

It is not surprising the parallel result is even faster than the sequential one as the information gathering even from reading it out from the cache is done in parallel and no resource congestion exists at the scale we use for our example.

Let us now compare the true time between sequential and parallel execution. Simply modify the code in the * line and replace the loop accordingly:

```
Sequential 12.681866169 s
Parallel    6.51530909538 s
```

Thus we see two interesting performance improvements

First, the performance improvement for running the queries in parallel. Second, the improvement of retrieving the results from a cache. The later is important if we have many user on the system executing the same command.

The lesson we learn is that clouds must make use of execution parallelism as well as addressing reuse of repeated results.

8.4 Exercises

1. Is pdsh installed? Where
2. Return the hostname of the machines sierra, india and foxtrot via the fabric command

3. Execute the command qstat with fabric on sierra and india. If you have an account on bigred2, please try it also there
4. Run the cloudmesh Sequential and parallel program. Modify your cloudmesh file accordingly
5. Advanced: compare the performance of the cache backend between Mongodb and the use of RabbitMQ while switching RabbitMQ out with Redis in the Celery code.
6. Advanced: provide a documentation on how to run celery for this example on Redis.

CHAPTER

NINE

IAAS

9.1 OpenStack Clouds

OpenStack is an open-sourced, IaaS cloud computing platform founded by Rackspace Hosting and NASA, and used widely in industry. Some of the modules that it provides to users includess compute, storage and network components allowing users to get easily stated in cloud computing.

Many clouds exist that have OpenStack as a foundation as part of the cloudservices offered to users. Some of them are free, some are commercial services.

9.1.1 OpenStack on FutureSystems

Note: Many of us use cloudmesh directly to access the various clouds. The interface cloudmesh provides is in regards to starting multiple virtual machines more convenient. Please try out the nova commands so you can appreciate what cloudmesh offers. For more information about cloudmesh see the Section *Cloudmesh*.

Note:

FutureSystems Portalname and Project ID For this example we assume you have set the shell variable :pink:PORTALNAME to your FutureSystems portal username. This can be done as follwows. Let us assume your portal name is *albert*. Than you can set it with:

```
$ export PORTALNAME=albert
```

If you execute the steps in this manual on india your india login name is the portalname, thus you can do:

```
$ export PORTALNAME=$USER
```

We also assume that you have a project id that you set to:

```
$ export PROJECTID=fg101
```

if it is the number 101.

Login

Currently FutureSystems OpenStack Havana installed on India. To use it you need to first log into india and prepare your Openstack credentials:

```
$ ssh $PORTALNAME@india.futuregrid.org
```

Setup OpenStack Environment

In case you like to use the shell command line tools you can load them with

```
$ module load novaclient
```

Creating the novarc file

An initial novarc file is currently created for you automatically and can be activated wih

```
$ source ~/.futuregrid/openstack_havana/novarc
```

In future this file will be created with the help of cloudmesh simplifying access to multiple heterogeneous clouds on FutureSystems.

List flavors

To list the flavors, please execute the following command

```
$ nova flavor-list
```

Everything is fine, if you see an output similar to

```
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------+-----------
| ID | Name      | Memory_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor | Is_Public | extra_spec
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------+-----------
| 1  | m1.tiny   | 512       | 0    | 0         |      | 1     | 1.0         | True      | {}
| 2  | m1.small  | 2048      | 20   | 0         |      | 1     | 1.0         | True      | {}
| 3  | m1.medium | 4096      | 40   | 0         |      | 2     | 1.0         | True      | {}
| 4  | m1.large  | 8192      | 80   | 0         |      | 4     | 1.0         | True      | {}
| 5  | m1.xlarge | 16384     | 160  | 0         |      | 8     | 1.0         | True      | {}
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------+-----------
```

If not your environment may not be set up correctly. Make sure that you follow the steps in this section and the account management section carefully.

List images

After you got the flavor list, you can list the current set of uploaded images with the nova image-list command:

```
$ nova image-list
```

You will see an output similar to:

```
+--------------------------------------+--------------------------+--------+--------+
| ID                                   | Name                     | Status | Server |
+--------------------------------------+--------------------------+--------+--------+
| 18c437e5-d65e-418f-a739-9604cef8ab33 | futuregrid/fedora-18     | ACTIVE |        |
| 1a5fd55e-79b9-4dd5-ae9b-ea10ef3156e9 | futuregrid/ubuntu-14.04  | ACTIVE |        |
+--------------------------------------+--------------------------+--------+--------+
```

Key management

Note: Make sure to check if you have not already created a key with the name at:

```
~/.ssh/$PORTALNAME-key
```

if so, please use another name. However, if you want to reuse the key, you certainly can do that. Also make sure the key is not already uploaded. This can be easily done in the following way:

```
$ nova keypait-list
```

To start a virtual machine you must first upload a key to the cloud:

```
$ nova keypair-add $PORTALNAME-key > ~/.ssh/$PORTALNAME-key
$ chmod 600 ~/.ssh/$PORTALNAME-key
$ nova keypair-list
+-----------------+-------------------------------------------------+
| Name            | Fingerprint                                     |
+-----------------+-------------------------------------------------+
| $PORTALNAME-key | ab:a6:63:82:dd:08:d3:bc:c0:21:56:4c:e2:bb:22:ac |
+-----------------+-------------------------------------------------+
```

Make sure you are not already having the key with that name in order to avoid overwriting it in the cloud. Thus be extra careful to execute this step twice. Often it is the case that you already have a key in your *~/.ssh* directory that you may want to use. For example if you use rsa, your key will be located at *~/.ssh/id_rsa.pub*.

Managing security groups

In the next step we need to make sure that the security groups allow us to log into the VMs. To do so we create the following policies as part of our default security policies. Not that when you are in a group project this may already have been done for you by another group member. We will add ICMP and port 22 on default group:

```
$ nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0
$ nova secgroup-add-rule default tcp 22 22 0.0.0.0/0
$ nova secgroup-list-rules default
```

Note: Most likely you will get some errors at this time as the definitions may already uploaded by default. simply ignore the errors and move on.

You will see the following output if everything went correctly:

```
+-------------+-----------+---------+-----------+--------------+
| IP Protocol | From Port | To Port | IP Range  | Source Group |
+-------------+-----------+---------+-----------+--------------+
| icmp        | -1        | -1      | 0.0.0.0/0 |              |
| tcp         | 22        | 22      | 0.0.0.0/0 |              |
+-------------+-----------+---------+-----------+--------------+
```

Booting an image

To boot an instance you simply can now use the command:

```
$ nova boot --flavor m1.small \
            --image "futuregrid/ubuntu-14.04" \
            --key_name $PORTALNAME-key $PORTALNAME-001
```

Please note that the last parameter is a "label" for the VM and we recommend thst you use a unique label. If everything went correctly, you will see an output similar to:

```
+-----------------------------+--------------------------------------+
| Property                    | Value                                |
+-----------------------------+--------------------------------------+
| status                      | BUILD                                |
| updated                     | 2013-05-15T20:32:03Z                 |
| OS-EXT-STS:task_state       | scheduling                           |
| key_name                    | $PORTALNAME-key                      |
| image                       | futuregrid/ubuntu-14.04              |
| hostId                      |                                      |
| OS-EXT-STS:vm_state         | building                             |
| flavor                      | m1.small                             |
| id                          | e15ad5b6-c3f0-4c07-996c-3bbe709a63b7 |
| security_groups             | [{u'name': u'default'}]              |
| user_id                     | 3bd2d773911c4502982e5c2cd81437f7     |
| name                        | vm001                                |
| adminPass                   | KgiKjek99dgk                         |
| tenant_id                   | b7ea98db7b3449b184b58d28e80c7541     |
| created                     | 2013-05-15T20:32:03Z                 |
| OS-DCF:diskConfig           | MANUAL                               |
| metadata                    | {}                                   |
| accessIPv4                  |                                      |
| accessIPv6                  |                                      |
| progress                    | 0                                    |
| OS-EXT-STS:power_state      | 0                                    |
| OS-EXT-AZ:availability_zone | None                                 |
| config_drive                |                                      |
+-----------------------------+--------------------------------------+
```

List running images

To check if your instance is active you can repeatedly issue the list command and monitor the Status field in the table:

```
$ nova list
```

```
+-------------+----------------+--------+---------------------+
| ID          | Name           | Status | Networks            |
+-------------+----------------+--------+---------------------+
| e15 ... 3b7 | $PORTALNAME-001| ACTIVE | private=10.35.23.18 |
+-------------+----------------+--------+---------------------+
```

Once it has changed from for example BUILD to ACTIVE, you can log in. Pleas use the IP address provided under networks. Note that the first address is private and can not be reached from outside india:

```
$ ssh -l ubuntu -i ~/.ssh/$PORTALNAME-key 10.35.23.18
```

If you see a warning similar to:

```
Add correct host key in ~/.ssh/known_hosts to get rid of this message.
Offending key in ~/.ssh/known_hosts:3
```

you need to delete the offending host key from ~/.ssh/known_hosts.

Use block storage

You can create a block storage with the volume-create command. A volume is useful as you can store data in it and associate that particular volumen to a VM. Hence, if you delete the VM, your volume and the data on it is still there to be reused. To create one 1G volume you can do

```
$ nova volume-create 1 --display-name $PORTALNAME-vol-001
```

To more conveniently identify the image we also specified a displayname. Please chose a uinque name so you can identify the volume more easily.

To list the volumes you can use:

```
$ nova volume-list
+--------------+-----------+---------------------+------+-------------+-------------+
| ID           | Status    | Display Name        | Size | Volume Type | Attached to |
+--------------+-----------+---------------------+------+-------------+-------------+
| 6d0d ... abc | available | $PORTALNAME-vol-001 |  1   | None        |             |
+--------------+-----------+---------------------+------+-------------+-------------+
```

To attach the volume to your instance you can use the volume-attach subcommand. Let us assume we like to attache it as "/dev/vdb", than you can use the command::

```
$ nova volume-attach $PORTALNAME-001 6d0d8285-xxxx-xxxx-xxxx-xxxxxxxxxabc "/dev/vdb"
```

Hint: Hint

$PORTALNAME-001 refers to the name of the VM that we have created earlier with the boot command.

Next, let us login to your instance, make filesystem and mount it. Here's an example, mounting on /mnt:

```
$ ssh -l ubuntu -i ~/.ssh/$PORTALNAME-key 10.35.23.18
ubuntu@$PORTALNAME-001:~$ sudo su -
root@$PORTALNAME-001:~# mkfs.ext4 /dev/vdb
root@$PORTALNAME-001:~# mount /dev/vdb /mnt
root@$PORTALNAME-001:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/vda1        20G  2.1G   17G  11% /
none            4.0K     0  4.0K   0% /sys/fs/cgroup
udev            998M  8.0K  998M   1% /dev
tmpfs           201M  236K  201M   1% /run
none            5.0M     0  5.0M   0% /run/lock
none           1002M     0 1002M   0% /run/shm
none            100M     0  100M   0% /run/user
/dev/vdb        4.8G   23M  0.8G   1% /mnt
```

When you want to detach it, unmount /mnt first, go back to indias's login node and execute volume-detach:

```
root@$PORTALNAME-001:~# umount /mnt
root@$PORTALNAME-001:~# exit
ubuntu@$PORTALNAME-001:~$ exit

$ nova volume-detach $PORTALNAME-001 6d0d8285-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

Set up external access to your instance

So far we only used the internal IP address, but you can also assign an external address, so that you can log in from other machines than india. Firts, Create an external ip address with:

```
$ nova floating-ip-create

+-----------------+-------------+----------+------+
| Ip              | Instance Id | Fixed Ip | Pool |
+-----------------+-------------+----------+------+
```

```
| 198.202.120.193 | None        | None     | nova |
+-----------------+-------------+----------+------+
```

Next, put it on your instance with:

```
$ nova add-floating-ip $PORTALNAME-001 198.202.120.193
$ nova floating-ip-list
```

```
+-----------------+--------------------------------------+-------------+------+
| Ip              | Instance Id                          | Fixed Ip    | Pool |
+-----------------+--------------------------------------+-------------+------+
| 198.202.120.193 | c0bd849a-221a-4e53-bf7b-7097541a9bcc | 10.35.23.20 | nova |
+-----------------+--------------------------------------+-------------+------+
```

Now you should be able to ping and ssh from external and can use the given ip address.

Make a snapshot of an instance

To allow snapshots, you must use the following convention:

- use your project number fg### in the prefix of your snapshot name followed by a /
- If needed you can also add your username as a prefix in addition to the project number. Replace the $PORTALNAME with the username of your FutureSystems account.

Let us assume your project is fg101 and you want to save the image with by reminding you it was a my-ubuntu-01 image you want to key. Than you can issue on india the following command:

```
$ nova image-create $PORTALNAME-001 fg101/$PORTALNAME/my-ubuntu-01
$ nova image-list
+--------------+--------------------------------+--------+--------------+
| ID           | Name                           | Status | Server       |
+--------------+--------------------------------+--------+--------------+
| 18c43 ... 33 | futuregrid/fedora-18           | ACTIVE |              |
| 1a5fd ... e9 | futuregrid/ubuntu-14.04        | ACTIVE |              |
| f4337 ... 44 | fg101/$PORTALNAME/my-ubuntu-01 | ACTIVE | c0bd ... bcc |
+--------------+--------------------------------+--------+--------------+
```

If you want to download your customized image, you can do it with this:

```
$ glance image-download --file "my-ubuntu-01.img" "fg101/$PORTALNAME/custom-ubuntu-01"
```

Warning: Please note that images not following this convention may be deleted without warning. Also ifyou do no longer need an image, please remove it.

Automate some initial configuration

You may want to install some packages into the image, enable root, and add ssh authorized_keys. With the OpenStack cloud-init such steps can be simplified.

Create a file(mycloudinit.txt) containing these lines:

```
#cloud-config

# Enable root login.
disable_root: false
```

```
# Install packages.
packages:
- apt-show-versions
- wget
- build-essential

# Add some more ssh public keys.
ssh_authorized_keys:
- ssh-rsa AAAfkdfeiekf....fES7060rb myuser@s1
- ssh-rsa AAAAAAkgeig78...skdfjeigi myuser@myhost
```

Now boot your instance with –user-data mycloudinit.txt like this:

```
$ nova boot --flavor m1.small \
            --image "futuregrid/ubuntu-14.04" \
            --key_name $PORTALNAME-key \
            --user-data mycloudinit.txt $PORTALNAME-002
```

You should be able to login to $PORTALNAME-002 as root, and the added packages are installed.

Get the latest version of Ubuntu Cloud Image and upload it to the OpenStack

Note: We will try to provide the latest images. E.g., currently in india openstack

the ubuntu 14.04 image is officially available under name: futuregrid/ubuntu-14.04. So usually you can skip this section to simply use the one provided officially.

Several versions of Ubuntu cloud images are available at http://cloud-images.ubuntu.com/. Choose the version you want and download the file name with ******-cloudimg-amd64-disk1.img. For example, downloading Ubuntu-14.04 is done like this:

```
$ wget https://cloud-images.ubuntu.com/trusty/current/trusty-server-cloudimg-amd64-disk1.img
```

If you need a different version, please adapt the link accordingly. You can upload the image with the glance client like this:

```
$ glance image-create \
       --name fg101/$PORTALNAME/myimages/ubuntu-14.04 \
       --disk-format qcow2 \
       --container-format bare \
       --file trusty-server-cloudimg-amd64-disk1.img
```

Now your new image is listed on `nova image-list` and will be available when the status become “ACTIVE”.

Delete your instance

You can delete your instance with:

```
$ nova delete $PORTALNAME-002
```

Please do not forget to also delete your 001 vm if you no longer need it.

How to change your password

1. Sometimes, users accidentally send password to a collaborator/support for debugging, and then regret. When you put yourself in the situation by mistake, don’t worry. Just use keystone client and reset your password with:

```
$ keystone password-update
```

Remember, you will also need to change it in your novarc. This can be achieved by either editing your novarc file directly, or by editing the file ~/.futuregrid/cloudmesh.yaml and recreating your novarc file.

Things to do when you need Euca2ools or EC2 interfaces

Even though the nova client and protocols will provide you with more advanced features, some users still want to access OpenStack with EC2 compatible tools. We recommend against this and recommend instead that you use *nova*. One such tool using eucarc files is euca2tools. We explain briefly how you can access them.

1. Create a directory for putting eucarc, and create pk.pem, cert.pem and cacert.pem:

   ```
   cd ~/.futuregrid/openstack_havana
   nova x509-create-cert
   nova x509-get-root-cert
   ls -la
   ```

2. Create EC2_ACCESS_KEY and EC2_SECRET_KEY:

   ```
   keystone ec2-credentials-create
   ```

3. Create the file calle *~/.futuregrid/openstack_havana/eucarc* and put your EC2_ACCESS_KEY and EC2_SECRET_KEY that you obtained from the previous command into this file:

   ```
   export NOVA_KEY_DIR=$(cd $(dirname ${BASH_SOURCE[0]}) && pwd)
   export EC2_ACCESS_KEY="Your EC2_ACCESS_KEY"
   export EC2_SECRET_KEY="Your EC2_SECRET_KEY"
   export EC2_URL="http://i57r.idp.iu.futuregrid.org:8773/services/Cloud"
   export S3_URL="http://i57r.idp.iu.futuregrid.org:3333"
   export EC2_USER_ID=11
   export EC2_PRIVATE_KEY=${NOVA_KEY_DIR}/pk.pem
   export EC2_CERT=${NOVA_KEY_DIR}/cert.pem
   export NOVA_CERT=${NOVA_KEY_DIR}/cacert.pem
   export EUCALYPTUS_CERT=${NOVA_CERT}
   alias ec2-bundle-image="ec2-bundle-image --cert ${EC2_CERT} --privatekey ${EC2_PRIVATE_KEY} --us
   alias ec2-upload-bundle="ec2-upload-bundle -a ${EC2_ACCESS_KEY} -s ${EC2_SECRET_KEY} --url ${S3_
   ```

4. Confirm if euca2ools works:

   ```
   module load euca2ools/3.1.0
   source ~/.futuregrid/openstack_havana/eucarc
   euca-describe-images
   euca-describe-instances
   ```

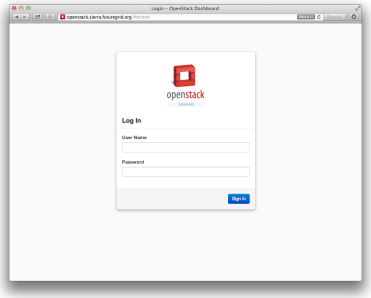
Note: Here's our known issues on using euca2ools or ec2 interface.

- euca-upload-bundle with Boto 2.25.0 fails with "S3ResponseError: 404 Not Found".
- tagging function such as euca-create-tags, euca-describe-tags fail with "InvalidRequest: The request is invalid."

Horizon GUI

Horizon is a graphical user interface/dashbooard for OpenStack. For starting up VMs and stoping them by hand horizon may be a good mechanism to manage your Virtual machines. We have currently the following horizon deployments available. However, please note that on Alamo an older version of Openstack is run.

Table 9.1: Horizon endpoints

Image	Version	Machine	Protocol	Description
	Havana	India	Native OpenStack	India offers a Graphical user interface to access OpenStack. For those interested in only managing a few images this may be a good way to start. The link to the GUI is https://openstack-h.india.futuregrid.org/horizon The password can be found by following the method discussed above.

Screencasts

This series of screencasts will walk you through the various ways on how you can use OpenStack on FutureSystems. This includes the following:

- using openstack client command line tools to
  - start, stop, assign ips, and query virtual machines
  - list images and flavors
  - to create security groups for login
  - to log in to your virtual machine while using a key
- using the openstack horizon interface

Video	Length	Titles of the Lessons	Description of the Lessons
xRVJ-fOaR23w	11:55 min	Using OpenStack command line tools	This lesson explains you how to use the OpenStack Commandline tools on the FutureSystems cluster called sierra.futuregrid.org. For written material, see section *OpenStack on FutureSystems*. **Warning:** please replace sierra with india.
JkNl-WAUlxF0	8:30 min	Using OpenStack horizon GUI	This lesson explains you how to use the OpenStack Horizon to access the FutureSystems OpenStack IaaS framework on sierra.futuregrid.org. For written material, see section *Horizon GUI*. **Warning:** please replace sierra with india.

Excersises

1. Create a VM on india and login
2. Create a volume and attach it to the vm
3. Read up on the openstack web page what vilumes are for.
4. Log into horizon and explore the interface. Start up a VM, create a volume and attach it to the VM. Assign a public ip and log in.

CHAPTER

TEN

PAAS

10.1 Using Hadoop in FutureSystems

Screencast

A screenncast of a subset of the information presented her is available at  PC8h1CtVzH4.

We have various platforms that support Hadoop on FutureSystems. MyHadoop is probably the easiest solution offered for you. It provides the advantage that it is integrated into the queuing system and allows hadoop jobs to be run as batch job. This is of especial interest for classes that may run quickly out of resources if every students wants to run their hadoop application at the same time.

10.1.1 Running Hadoop as a Batch Job using MyHadoop

MapReduce is a programming model developed by Google. Their definition of MapReduce is as follows: "MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key." For more information about MapReduce, please see the Google paper here.

The Apache Hadoop Project provides an open source implementation of MapReduce and HDFS (Hadoop Distributed File System).

This tutorial illustrates how to run Apache Hadoop thru the batch systems on FutureSystems using the MyHadoop tool.

myHadoop on FutureSystems

MyHadoop is a set of scripts that configure and instantiate Hadoop as a batch job.

myHadoop 0.20.2 is currently installed on Alamo, Hotel, India, and Sierra FutureSystems systems.

Login into a machine tha has myHadoop installed

To run the example we need to firts log into a FutureSystems system that has myHadoop available. In this tutorial, we are executing from the sierral machine:

```
$ ssh portalname@sierra.futuregrid.org
```

Note that this also works on other FutureSystems machines such as india.

This machine accepts SSH public key and One Time Password (OTP) logins only. If you do not have a public key set up, you will be prompted for a password. This is *not* your FutureSystems password, but the One Time Password generated from your OTP token. Do not type your FutureSystems password, it will not work. If you do not have a token or public key, you will not be able to login. The portalname is your account name that allows you to log into the FutureSystems portal.

Load the needed modules

Next, we need to load the myHadoop module. On some FutureSystems systems, you may also need to load the "torque" module as well if qstat is not already in your environment:

```
$ module load myhadoop
SUN Java JDK version 1.6.0 (x86_64 architecture) loaded
Apache Hadoop Common version 0.20.203.0 loaded
myHadoop version 0.2a loaded
```

Before we can submit it we still need to load the module java as Haddop relies on java:

```
module add java
```

Run the Example

To run the example we need to create a script to tell the queing system how to run it. We are providing the following script that you can store in a file pbs-example.sh.

You can paste and copy it from here, or just copy it via:

```
cp $MY_HADOOP_HOME/pbs-example.sh .
```

This script includes information about which queue hadoop should be run in. To find out more about the queuing system, please see the HPC services section. The pbs-example.sh script we use in this example looks as follows:

```
#!/bin/bash

#PBS -q batch
#PBS -N hadoop_job
#PBS -l nodes=4:ppn=8
#PBS -o hadoop_run.out
#PBS -j oe
#PBS -V

module add java

### Run the myHadoop environment script to set the appropriate variables
#
# Note: ensure that the variables are set correctly in bin/setenv.sh
. /N/soft/myHadoop/bin/setenv.sh

#### Set this to the directory where Hadoop configs should be generated
# Don't change the name of this variable (HADOOP_CONF_DIR) as it is
# required by Hadoop - all config files will be picked up from here
#
```

```
# Make sure that this is accessible to all nodes
export HADOOP_CONF_DIR="${HOME}/myHadoop-config"

#### Set up the configuration
# Make sure number of nodes is the same as what you have requested from PBS
# usage: $MY_HADOOP_HOME/bin/pbs-configure.sh -h
echo "Set up the configurations for myHadoop"
# this is the non-persistent mode
$MY_HADOOP_HOME/bin/pbs-configure.sh -n 4 -c $HADOOP_CONF_DIR
# this is the persistent mode
# $MY_HADOOP_HOME/bin/pbs-configure.sh -n 4 -c $HADOOP_CONF_DIR -p -d /oasis/cloudstor-group/HDFS
echo

#### Format HDFS, if this is the first time or not a persistent instance
echo "Format HDFS"
$HADOOP_HOME/bin/hadoop --config $HADOOP_CONF_DIR namenode -format
echo

#### Start the Hadoop cluster
echo "Start all Hadoop daemons"
$HADOOP_HOME/bin/start-all.sh
#$HADOOP_HOME/bin/hadoop dfsadmin -safemode leave
echo

#### Run your jobs here
echo "Run some test Hadoop jobs"
$HADOOP_HOME/bin/hadoop --config $HADOOP_CONF_DIR dfs -mkdir Data
$HADOOP_HOME/bin/hadoop --config $HADOOP_CONF_DIR dfs -copyFromLocal $MY_HADOOP_HOME/gutenberg Data
$HADOOP_HOME/bin/hadoop --config $HADOOP_CONF_DIR dfs -ls Data/gutenberg
$HADOOP_HOME/bin/hadoop --config $HADOOP_CONF_DIR jar $HADOOP_HOME/hadoop-0.20.2-examples.jar wordcou
$HADOOP_HOME/bin/hadoop --config $HADOOP_CONF_DIR dfs -ls Outputs
$HADOOP_HOME/bin/hadoop --config $HADOOP_CONF_DIR dfs -copyToLocal Outputs ${HOME}/Hadoop-Outputs
echo

#### Stop the Hadoop cluster
echo "Stop all Hadoop daemons"
$HADOOP_HOME/bin/stop-all.sh
echo

#### Clean up the working directories after job completion
echo "Clean up"
$MY_HADOOP_HOME/bin/pbs-cleanup.sh -n 4 -c $HADOOP_CONF_DIR
echo
```

Details of the Script

Let us examine this script in more detail. In the example script, a temporary directory to store Hadoop configuration files is specified as ${HOME}/myHadoop-config:

```
#### Set this to the directory where Hadoop configs should be generated
# Don't change the name of this variable (HADOOP_CONF_DIR) as it is
# required by Hadoop - all config files will be picked up from here
#
# Make sure that this is accessible to all nodes
export HADOOP_CONF_DIR="${HOME}/myHadoop-config"
```

The pbs-example.sh script runs the "wordcount" program from the hadoop-0.20.2-examples.jar. There is sample text

data from the Project Gutenberg website located a $MY_HADOOP_HOME/gutenberg:

```
$ ls $MY_HADOOP_HOME/gutenberg
1342.txt.utf8
```

The following lines in the script create a data directory in HDFS. This directory is specified in $MY_HADOOP_HOME/bin/setenv.sh. To activate the environment, pleas execute:

```
source $MY_HADOOP_HOME/bin/setenv.sh
```

The next lines in the script will copy over the gutenberg data, executes the Hadoop job, and then copies the output back your ${HOME}/Hadoop-Outputs directory.

```
#### Run your jobs here
echo "Run some test Hadoop jobs"
$HADOOP_HOME/bin/hadoop --config $HADOOP_CONF_DIR dfs -mkdir Data
$HADOOP_HOME/bin/hadoop --config $HADOOP_CONF_DIR dfs -copyFromLocal $MY_HADOOP_HOME/gutenberg Data
$HADOOP_HOME/bin/hadoop --config $HADOOP_CONF_DIR dfs -ls Data/gutenberg
$HADOOP_HOME/bin/hadoop --config $HADOOP_CONF_DIR jar $HADOOP_HOME/hadoop-0.20.2-examples.jar wordcou
$HADOOP_HOME/bin/hadoop --config $HADOOP_CONF_DIR dfs -ls Outputs
$HADOOP_HOME/bin/hadoop --config $HADOOP_CONF_DIR dfs -copyToLocal Outputs ${HOME}/Hadoop-Outputs
```

Submission of the Hadoop job

Now submit the pbs-example.sh script to Hotel:

```
$ qsub $MY_HADOOP_HOME/pbs-example.sh
40256.svc.uc.futuregrid.org
```

The job will take about 5 minutes to complete. To monitor its status, type ‘qstat’. The “R” means the job is running:

```
$ qstat
Job id                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
40256.svc                  hadoop_job       albert                 0 R batch
```

When it is done, the status of the job will be “C” meaning the job has completed (or it will no longer be displayed in qstat output). You should see a new hadoop_run.out file and an “Hadoop-Outputs” directory

```
$ qstat
Job id                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
40256.svc                  hadoop_job       albert         00:00:05 C batch
$ ls
Hadoop-Outputs hadoop_run.out
```

View results of the word count operation:

```
$ head Hadoop-Outputs/part-r-00000
"'After    1
"'My   1
"'Tis  2
"A 12
"About 2
"Ah!   2
"Ah!" 1
"Ah,   1
"All   2
"All!  1
```

Now to run you own custom Hadoop job, make a copy of the $MY_HADOOP_HOME/pbs-example.sh script and modify the lines described in Step 7.

Persistent Mode

The above example copies input to local HDFS scratch space you specified in $MY_HADOOP_HOME/bin/setenv.sh, runs MapReduce, and copies output from HDFS back to your home directory. This is called non-persistent mode and is good for small amounts of data. Alternatively, you can run in persistent mode which is good if you have access to a parallel file system or have a large amount of data that will not fit in scratch space. To enable persistent mode, follow the directions in pbs-example.sh.

Customizing Hadoop Settings

To modify any of the Hadoop settings like maximum_number_of_map_task, maximum_number_of_reduce_task, etc., make you own copy of myHadoop and customize the settings accordingly. For example:

1. Copy the $MY_HADOOP_HOME directory to your home directory:

   ```
   $ cp -r $MY_HADOOP_HOME $HOME/myHadoop
   ```

2. Then edit $HOME/myHadoop/pbs-example.sh and on line 16, replace it with:

   ```
   . ${HOME}/myHadoop/bin/setenv.sh
   ```

3. Similarly edit $HOME/myHadoop/bin/setenv.sh and on line 4, replace it with:

   ```
   export MY_HADOOP_HOME=$HOME/myHadoop
   ```

4. Customize the settings in the Hadoop files as needed in $HOME/myHadoop/etc
5. Submit your copy of pbs-example.sh:

   ```
   $ qsub $HOME/myHadoop/pbs-example.sh
   ```

Using a Different Installation of Hadoop

If you would like to use a different version of my Hadoop or have customized the Hadoop code in some way, you can specify a different installation of Hadoop by redefining the HADOOP_HOME variable after $MY_HADOOP_HOME/bin/setenv.sh is called within your own copy of pbs-example.sh:

```
### Run the myHadoop environment script to set the appropriate variables
#
# Note: ensure that the variables are set correctly in bin/setenv.sh
. /opt/myHadoop/bin/setenv.sh
export HADOOP_HOME=${HOME}/my-custom-hadoop
```

References

- Much of this information is copied from The MyHadoop Installation Instructions
- A screenncast of a subset of the information presented her is available at  PC8h1CtVzH4.

10.2 Deploying a Hadoop Cluster on India OpenStack

10.2.1 Introduction

This tutorial assumes you already have used OpenStack and know how to create multiple virtual machine images. To build a distributed Hadoop cluster, you will need at least two VMs for nodes, though more than two are welcome. Much of the functionality described here will be available via Cloudmesh, but this document explains how to perform these tasks manually, to help you better understand the requirements and process of building a cluster.

10.2.2 Cluster Preparation

Prior to deploying Hadoop, your nodes must be able to communicate. This will require changes to the */etc/hosts* configuration file, and creation and sharing of SSH keys among the nodes of the cluster.

Add lines to the end of your */etc/hosts* file, one line for each node in the cluster, listing the IP address, fully qualified host name, and an alias for the host to be used by Hadoop. Here is an example of an */etc/host* file with six nodes added. Of course you will use the proper IP addresses and host names for your VMs. You can use the same hosts file for every node in your cluster:

```
127.0.0.1 localhost

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts

# lines added for Hadoop cluster
10.39.1.46 smccaula-101.novalocal hadoop1
10.39.1.47 smccaula-102.novalocal hadoop2
10.39.1.55 smccaula-103.novalocal hadoop3
10.39.1.56 smccaula-104.novalocal hadoop4
10.39.1.57 smccaula-105.novalocal hadoop5
10.39.1.45 smccaula-106.novalocal hadoop6
```

Your nodes will also SSH authentication to communicate. For each node, you will need to create a pair of SSH keys (as root). The following command will create a key pair in */root/.ssh*:

```
ssh-keygen -t rsa -P ""
```

You will need to append the public key created (default will be id_rsa.pub) to the authorized_keys file of each node. You can do this by downloading the keys from one host and uploading them to another, or by copying and pasting them from an editor program in your terminal. If copying and pasting, be sure all characters are copied. Given you have moved a public key to another host, you can append it to authorized_key as follows (again, this is as root and the files are in */root/.ssh*):

```
cat hadoop2.pub >> /root/.ssh/authorized_keys
```

Test this by verifying that you can SSH from node to node in either direction. From this point, implementing a multi-node cluster is very similar to building a single node pseudo-cluster. The main difference is that we will establish some division of labor among the nodes. There are three functions we need to fill:

NameNode for the HDFS file system:

- Keeps track of all files and on which nodes they are stored

ResourceManager for the YARN resource negotiator:

- Manages cluster resources and applications

DataNode(s) for the HDFS file system

- Stores data files and makes them available to client applications

cm cluster create: a convenient way to create a cluster of VMs

Cloudmesh provides a convenient way to create such a cluster of VMs (each of them can log into all others). Please follow the following steps:

Important: Make sure you update your cloudmesh and the cloudmesh server is running. Open a terminal, execute the following commands, modify the values of the options according to your own environment and needs. Also you may execute these in cloudmesh CLI 'cm'.

---

1 select cloud to work on, e.g.:

```
cm "cloud select india"
```

2 activate the cloud, e.g.:

```
cm "cloud on india"
```

3 set the default key to start VMs, e.g.:

```
cm "key default test-key"
```

4 set the start name of VMs, which is prefix and index, e.g.:

```
cm "label --prefix=test --id=1"
```

5 set image of VMs, e.g.:

```
cm "default image --name=futuregrid/ubuntu-14.04"
```

6 set flavor of VMs, e.g.:

```
cm "default flavor --name=m1.small"
```

Then you may start the cluster with command 'cluster create' by providing the following values:

–count: specify amount of VMs in the cluster

–group: specify a group name of the cluster, make sure it's UNIQUE

–ln: login name for VMs, e.g. ubuntu e.g.:

```
cm "cluster create --count=3 --group=test --ln=ubuntu"
```

You may also provide cloud name, flavor or image in the command if you don't want to pre-set them. e.g.:

```
cm "cluster create --count=3 --group=test0 --ln=ubuntu --cloud=india --flavor=m1.small --image=future
```

to list the VMs you just created:

```
cm "vm list --refresh --group=test"
```

10.2.3 Deploying Hadoop

We will have to decide on the architecture of our cluster before proceeding. In practice, clusters can be tens of thousands of nodes, but our cluster will be a handful of nodes. For our example, we will combine all the management functions on one node, and make the rest datanodes.

We will use Chef to install the Hadoop software, and configure our nodes, calling different recipes for the manager and worker nodes. In addition to Hadoop, we will install Oracle Java, as that is Hadoop's preferred version of Java. The Apt and Yum cookbooks are also downloaded as they are required by the Hadoop recipe. First we need to install Chef and download the required cookbooks from the Chef repository. As root, and in the /home/ubuntu directory, these commands will do that:

```
curl -L https://www.opscode.com/chef/install.sh | bash
wget http://github.com/opscode/chef-repo/tarball/master
tar -zxf master
mv opscode-chef-repo* chef-repo
rm master
cd chef-repo/
mkdir .chef
echo "cookbook_path [ '/home/ubuntu/chef-repo/cookbooks' ]" > .chef/knife.rb
cd cookbooks
knife cookbook site download java
knife cookbook site download apt
knife cookbook site download yum
knife cookbook site download hadoop
tar -zxf java*
tar -zxf apt*
tar -zxf yum*
tar -zxf hadoop*
rm *.tar.gz
```

There are four files we will need to create to store our preferences. These will need slight customization based on your host names and your desired configuration. In */home/ubuntu/chef-repo/roles* create java.rb for our Java preferences. We request Oracle Java version 6, and ask to have the *$JAVA_HOME* environment variable set automatically:

```
name "java"
description "Install Oracle Java"
default_attributes(
  "java" => {
    "install_flavor" => "oracle",
    "jdk_version" => "6",
    "set_etc_environment" => true,
    "oracle" => {
      "accept_oracle_download_terms" => true
    }
  }
)
run_list(
  "recipe[java]"
)
```

In */home/ubuntu/chef-repo/roles* create hadoop.rb for our Hadoop preferences. These preferences will actually be the same whether we are installing a namenode or a datanode, we will just call a different recipe. Here we will pass the names of our HDFS and YARN manager nodes. In this example the manager node has an alias of hadoop1. If you named yours differently, change it here to match:

```
name "hadoop"
description "set Hadoop attributes"
default_attributes(
```

```
  "hadoop" => {
    "distribution" => "bigtop",
    "core_site" => {
      "fs.defaultFS" => "hdfs://hadoop1"
    },
    "yarn_site" => {
      "yarn.resourcemanager.hostname" => "hadoop1"
    }
  }
)
run_list(
  "recipe[hadoop]"
)
```

In */home/ubuntu/chef-repo create solo.rb* to store locations and instructions for Chef to use:

```
file_cache_path "/home/ubuntu/chef-solo"
cookbook_path "/home/ubuntu/chef-repo/cookbooks"
role_path "/home/ubuntu/chef-repo/roles"
verify_api_cert true
```

Finally, in */home/ubuntu/chef-repo* create solo.json for the specific instructions to Chef on what to install. This is the only file that will change between a manager and worker node installation. Both versions are shown below. Remember that you could configure differently, the HDFS namenode and YARN resoursemanager could be on different nodes, and the namenode and resoursemanager nodes could also be datanodes if desired. You may want to install and initialize your manager node prior to creating your worker node.

For the manager node:

```
{
  "run_list": [ "role[java]", "recipe[java]", "role[hadoop]", "recipe[hadoop::hadoop_hdfs_namenode]",
   "recipe[hadoop::hadoop_yarn_nodemanager]", "recipe[hadoop::hadoop_yarn_resourcemanager]" ]
}
```

For the worker node:

```
{
  "run_list": [ "role[java]", "recipe[java]", "role[hadoop]",  "recipe[hadoop::hadoop_hdfs_datanode]"
}
```

Repeat the worker installation for as many nodes as are available. At this point your cluster is deployed and awaiting initialization.

10.2.4 Initializing and Testing

On the namenode only, we will have to initialize the file system. First check the status of all services and stop any that are running. Don't worry about services not installed on this node:

```
service hadoop-hdfs-namenode status
service hadoop-hdfs-datanode status
service hadoop-yarn-resourcemanager status
service hadoop-yarn-nodemanager status
```

To initialize the namenode, run:

```
/etc/init.d/hadoop-hdfs-namenode init
```

Restart any services installed on the node. There is one more initialization step required on the namenode, to create a default directory structure:

```
/usr/lib/hadoop/libexec/init-hdfs.sh
```

When these initialization steps are complete, and all the appropriate services are running on each node, the Hadoop cluster will be operational and ready to run jobs.

CHAPTER

ELEVEN

HPC

11.1 HPC

11.1.1 Login Nodes

Several of the clusters have High Performance Computing (HPC) services installed. Access to them is provided via a Linux Login node for each of the clusters on which these services are installed.

To access the login nodes you need a FG resource account and an SSH public key you have uploaded to FutureGrid (this process is described in the section about *Account and Project Management*. After you are part of a valid project and have a FutureGrid account, you can log into the FutureGrid resources with ssh. Multiple systems are accessed through the follong node:

```
india.futuregrid.org
```

The systems include india, bravo, delta, echo

To log into xray, please use:

```
xray.futuregrid.org
```

For example, assume your portalname is "portalname", than you can login to sierra as follows:

```
$ ssh portalname@india.futuregrid.org
Welcome to india.futuregrid.org
Last login: Thu Aug 12 19:19:22 2010 from ....
```

SSH Add

Sometimes you may wish to log in repeatedly in other machines while using a cached password. To do that you can use ssh agent and ssh add. First start the agent:

```
eval `ssh-agent`
```

Then add your key with:

```
ssh-add
```

Follow the instructions on the screen. Thus before you ssh in, you may want to use ssh agent. This way you do not have to repeatedly type in your key password.

SSH Config

Also you may want to setup your ~/.ssh/config file to create shortcut for the username and hosts on which you want to log in. Let us assume your username is albert, then add the following lines in the .ssh/config file:

```
Host india
      Hostname india.futuregrid.org
      User albert
```

This will allow you to log into the machine just while typing in:

```
ssh india
```

Modules

The login nodes have the modules package installed. It provides a convenient tool to adapt your environment and enables you to activate different packages and services dependent on your specific needs. The Modules are utilized to let you dynamically control your environment. Modules allows you to load and unload packages and ensure a coherent working environment. This ensures that your $PATH, $LD_LIBRARY_PATH, $LD_PRELOAD, and other environment variables are properly set, and that you can access the programs and libraries you need. For additional information about the Modules package you can consult the manual page.

To display the list of available modules:

```
$ module avail
```

To display the list of currently loaded modules:

```
$ module list
```

To add and remove packages from your environment you can use the *module load* and *module unload* commands:

```
$ module load <package name>/<optional package version>
$ module unload <package name>/<optional package version>
```

The available command are listed in the next table:

Table 11.1: Module commands

Command	Description
module avail	List all software packages available on the system.
module avail package	List all versions of package available on the system
module list	List all packages currently loaded in your environment.
module load package/version	Add the specified version of the package to your environment
module unload package	Remove the specified package from your environment.
module swap package_A package_B	Swap the loaded package (package_A) with another package (package_B).
module show package	Shows what changes will be made to your environment (e.g. paths to libraries and executables) by loading the specified package.

Example - List the currently loaded modules on india after login:

```
$ module list

Currently Loaded Modulefiles:
  1) torque/2.4.8   2) moab/5.4.0
```

Example - list the avialable modules on india:

```
$ module avail

----------------- /opt/Modules/3.2.8/modulefiles/applications ------------------
R/2.11.1(default)      hpcc/1.3.1(default)    velvet/1.0.15
git/1.7.10             ncbi/2.2.23(default)   wgs/6.1
gromacs/4.0.7(default) soapdenovo/1.04

------------------- /opt/Modules/3.2.8/modulefiles/compilers -------------------
cmake/2.8.1(default)       java/1.6.0-i586
intel/10.1                 java/1.6.0-x86_64(default)
intel/11.1(default)

------------------- /opt/Modules/3.2.8/modulefiles/debuggers -------------------
null                       totalview/8.8.0-2(default)

------------------- /opt/Modules/3.2.8/modulefiles/libraries -------------------
intelmpi/4.0.0.028(default)  openmpi/1.4.3-intel
mkl/10.2.5.035(default)      otf/1.7.0(default)
openmpi/1.4.2(default)       unimci/1.0.1(default)
openmpi/1.4.3-gnu            vampirtrace/intel-11.1/5.8.2

---------------------- /opt/Modules/3.2.8/modulefiles/tools ----------------------
cinderclient/1.0.4(default)   moab/5.4.0(default)
cloudmesh/0.8(default)        myhadoop/0.2a
euca2ools/1.2                 novaclient/2.13.0(default)
euca2ools/1.3.1               precip/0.1(default)
euca2ools/2.0.2(default)      python/2.7(default)
genesisII/2.7.0               python/2.7.2
glanceclient/0.9.0(default)   torque/2.4.8(default)
keystoneclient/0.2.3(default) vim/7.2
marmot/2.4.0(default)
```

Example - load the default version of a module (in this case git):

```
$ module load git
```

Please note that for loading the default you do not have to specify the version number.

Filesystem Layout

***Home* directories:** Home directories are accessible through the $HOME shell variable which are located at */N/u/<username>*. This is where users are encouraged to keep source files, configuration files and executables. Users should not run code from their $HOME directories. Please note that this is an NFS file system, and may result in slower access for some applications. We also advise the users to provide external backup storage at their home institution or a code repository. For example, we recommend that you use git or svn to make sure you backup your changes to the code. Also make sure you backup your data. As a testbed, we do not guarantee data loss.

***Scratch* directories:** Scratch directories are located at different locations on the systems. To find out more about the file layout, please see the section *s-storage*

***System software* directories:** System software directories are located at */N/soft*. System and community software are typically installed here. Table *t-storage-mountpoint* provides a summary of the various mount points.

11.1.2 Message Passing Interface (MPI)

The *Message Passing Interface Standard (MPI)* is the *de facto* standard communication library for almost many HPC systems, and is available in a variety of implementations. It has been created through consensus of the MPI Forum, which has dozens of participating organizations, including vendors, researchers, software library developers, and users. The goal of the Message Passing Interface is to provide a portable, efficient, and flexible standard for programs using message passing. For more information about MPI, please visit:

- http://www.mpi-forum.org/
- http://www.mcs.anl.gov/research/projects/mpi/tutorial/
- http://www.open-mpi.org/

MPI Libraries

Several FutureGrid systems support MPI as part of their HPC services. An up to date status about it can be retrieved via our Inca status pages.

Table 11.2: MPI versions installed on FutureGrid HPC services

System	MPI version	Compiler	Infiniband Support	Module	
Bravo	OpenMPI 1.4.2	Intel 11.1	no	openmpi	
	OpenMPI 1.4.3	gcc 4.4.6	no	openmpi/1.4.3-gnu	
	OpenMPI 1.4.3	Intel 11.1	no	openmpi/1.4.3-intel	
	OpenMPI 1.5.4	gcc 4.4.6	no	openmpi/1.5.4-[gnu	intel]
India	OpenMPI 1.4.2	Intel 11.1	yes	openmpi	
Xray			N/A		

Loading the OpenMPI module adds the MPI compilers to your $PATH environment variable and the OpenMPI shared library directory to your $LD_LIBRARY_PATH. This is an important step to ensure that MPI applications will compile and run successfully. In cases where the OpenMPI is compiled with the Intel compilers loading the OpenMPI module will automatically load the Intel compilers as a dependency. To load the default OpenMPI module and associated compilers, just use:

```
$ module load openmpi
```

Compiling MPI Applications

To compile MPI applications, users can simply use the available mpi compile commands:

mpicc: To compile C programs with the CC/icc/gcc compilers

mpicxx: To compile c++ programs with CXX/icpc/g++ compilers

mpif90: To compile programs with F90/F77/FC/ifort/gfortran

To see in detail what these commands do you can add a *-show* as an option. Thus the following commands:

```
$ mpicc -show
$ mpicxx -show
$ mpif90 -show
```

will show you the detail of each of them. The resulting output can be used as a template to adapt compile flags in case the default settings are not suitable for you.

Assuming you have loaded the OpenMPI module into your environment, you can compile a simple MPI application easily by executing:

```
$ mpicc -o ring ring.c
```

Users MUST NOT run jobs on the login or headnodes. These nodes are reserved for editing and compiling programs. Furthermore running your commands on such nodes will not provide any useful information as you actually do not use the standard cluster node.

Batch Jobs

Once your MPI application is compiled, you run it on the compute nodes of a cluster via a batch processing. With the help of a batch processing services a job is run on the cluster without the users intervention via a job queue. The user does not have to worry much about the internal details of the job queue, but must provide the scheduler with some guidance about the job so it can be efficiently scheduled on the system.

To run jobs on resources with the HPC services, users must first activate their environment to use the job scheduler:

```
$ module load torque
```

A complete manual for the torque scheduler can be found in the Torque manual .

Next we need to create a script so that we can run the program on the cluster. We will be using our simple ring example to illustrate some of the parameters you need to adjust. Please save the following content to a file called ring.pbs.:

```
#! /bin/bash

# OPTIONS FOR THE SCRIPT
#PBS -M username@example.com
#PBS -N ring_test
#PBS -o ring_$PBS_JOBID.out
#PBS -e ring_$PBS_JOBID.err
#PBS -q batch
#PBS -l nodes=4:ppn=8
#PBS -l walltime=00:20:00


# make sure MPI is in the environment
module load openmpi

# launch the parallel application with the correct number of process
# Typical usage: mpirun -np <number of processes> <executable> <arguments>
mpirun -np 32 ring -t 1000

echo "Nodes allocated to this job: "
cat $PBS_NODEFILE
```

This file can be used to submit a job to the queueing system by calling the command:

```
qsub ring.pbs
```

In the job script, lines that begin with **#PBS** are directives to the job scheduler. You can disable any of these lines by adding an extra **#** character at the beginning of the line, as *##* is interpreted to be a comment. Common options include:

- -M: specify a mail address that is notified upon completion
- -N: To specify a job name
- -o: The name of the file to write stdout to
- -e: The name of the file to write stderr to

- -q: The queue to submit the job to
- -l: Resources specifications to execute the job

The first parameters are rather obvious, so let us focus on the *-q* option. Each batch service is configured with a number of queues that are targeting different classes of jobs to schedule them more efficiently. These queues can be switch on or off, be modified or new queues can be added to the system. It is useful to get a list of available queues on the system of where you would like to submit your jobs. You can also inspect which would be the most suitable queue to use for your purpose with the qstat command on the appropriate login node:

```
$ qstat -q
```

Currently we have the following queues:

HPC Job Queue Information:

Resource	Queue name	Default Wallclock Limit	Max Wallclock Limit	NOTES
india	batch	4 hours	24 hours	
	long	8 hours	168 hours	
	scalemp	8 hours	168 hours	restricted acce

Next we focus on the -l option that specifies the resources. The term:

```
nodes=4
```

means that we specify 4 servers on which we execute the job. The term:

```
ppn=8
```

means that we use 8 virtual processors per node, where a virtual processor is typically executed on a core of the server. Thus it is advisable not to exceed the number of cores per server. For some programs choosing the best performing number of servers and cores may be dependent on factors such as memory needs, IO access and other resource bounded properties. You may have to experiment with the parameters. To identify the number of servers and cores available please see Tables *Table: Compute Resources* and *Table: Compute Resource Details*. For example, India has 8 cores per node, thus 4 servers would provide you access to 32 processing units.

Often you may just want to have the stdout and stderr in one file, then you simply can replace the line with -e in it with:

```
#PBS -j oe
```

which simply means that you *join* stdout and stderr. Here j stands for join, o for stdout and e for stderr. In case you would like to have an e-mail sent to you based on the status of the job, you can add:

```
#PBS -m ae
```

to your script. It will send you a mail when the job aborts (indicated by a), or when the job ends (indicated by e).

11.1.3 Job Management

A list of all available scheduler commands is available from the Torque manual page. We describe next the use of some typical interactions to manage your jobs in the batch queue.

Job Submission

Once you have created a submission script, you can then use the qsub command to submit this job to be executed on the compute nodes:

```
$ qsub ring.pbs
20311.i136
```

The qsub command outputs either a job identifier or an error message describing why the scheduler would not accept your job. Alternatively, you can also use the msub command, which is very similar to the qsub command. For differences we ask you to consult the manual pages.

Job Deletion

Sometimes you may want to delete a job from the queue, which can be easily done with the qdel command, followed by the id:

```
$ qdel 20311
```

Job Monitoring

If your job is submitted successfully, you can track its execution using the qstat or showq commands. Both commands will show you the state of the jobs submitted to the scheduler. The difference is mostly in their output format.

showq: Divides the output into three sections: active jobs, eligible jobs, and blocked jobs:

```
$ showq
active jobs
------------------------
JOBID    USERNAME       STATE PROCS    REMAINING            STARTTIME
20311   yourusername       Running     16        3:59:59 Tue Aug 17 09:02:40
1 active job 16 of 264 processors in use by local jobs (6.06%)
                  2 of 33 nodes active (6.06%) eligible jobs
----------------------
JOBID    USERNAME       STATE PROCS    REMAINING            STARTTIME
0 eligible jobs blocked jobs
------------------------
JOBID    USERNAME       STATE PROCS    REMAINING            STARTTIME
0 blocked jobs
Total job: 1
```

Legend:

Active jobs: are jobs that are currently running on resources.

Eligible jobs: are jobs that are waiting for nodes to become available before they can run. As a general rule, jobs are listed in the order that they will be scheduled, but scheduling algorithms may change the order over time.

Blocked jobs: are jobs that the scheduler cannot run for some reason. Usually a job becomes blocked because it is requesting something that is impossible, such as more nodes than those which currently exist, or more processors per node than are installed.

qstat: provides a single table view, where the status of each job is added via a status column called S:

```
$ qstat
Job id                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
1981.i136                 sub19327.sub     inca             00:00:00 C batch
20311.i136                testjob          yourusername            0 R batch
```

Legend:

Job id: is the identifier assigned to your job.

Name: is the name that you assigned to your job.

User: is the username of the person who submitted the job.

Time: is the amount of time the job has been running.

S: shows the job state. Common job states are R for a running job, Q for a job that is queued and waiting to run, C for a job that has completed, and H for a job that is being held.

Queue: is the name of the job queue where your job will run.

If you are interested in only your job use grep:

```
$ qstat | grep 20311
```

Job Output

If you gave your job a name with the **#PBS -N <jobname>** directive in your job script or by specifying the job name on the command line, your job output will be available in a file named **jobname.o######**, where the ###### is the job number assigned by the job manager. You can type **ls jobname.o*** to see all output files from the same job name.

If you explicitly name an output file with the **#PBS -o <outfile>** directive in your job script or by specifying the output file on the command line, your output will be in the file you specified. If you run the job again, the output file will be overwritten.

If you don't specify any output file, your job output will have the same name as your job script, and will be numbered in the same manner as if you had specified a job name (**jobname.o######**).

11.1.4 Xray HPC Services

To log into the login node of xray please use the command:

```
ssh portalname@xray.futuregrid.org
```

Extensive documentation about the user environment of the Cray can be found at

- Cray XTTM Programming Environment User's Guide

For MPI jobs, use cc (pgcc). For best performance, add the xtpe-barcelona module:

```
% module add xtpe-module
```

Currently there is only one queue (batch) available to users on the Cray, and all jobs are automatically routed to that queue. You can use the same commands as introduced in the previous sections. Thus, to list the queues please use:

```
qstat -Q
```

To obtain details of running jobs and available processors, use the showq command:

```
/opt/moab/default/bin/showq
```

Submitting a Job on xray

To execute an MPI program on xray we use a special program called aprun in the submit script. Additionally we have some special resource specifications that we can pass along, such as mppwidth and mppnppn. An example is the following program that will use 16 processors on 2 nodes:

```
$ cat job.pbs
```

```
#! /bin/sh

#PBS -l mppwidth=16
#PBS -l mppnppn=8
#PBS -N hpcc-16
#PBS -j oe
#PBS -l walltime=7:00:00

#cd to directory where job was submitted from
cd $PBS_O_WORKDIR
export MPICH_FAST_MEMCPY=1
export MPICH_PTL_MATCH_OFF=1
aprun -n 16 -N 8 -ss -cc cpu hpcc

$ qsub job.pbs
```

The XT5m is a 2D mesh of nodes. Each node has two sockets, and each socket has four cores. The batch scheduler interfaces with a Cray resource scheduler called APLS. When you submit a job, the batch scheduler talks to ALPS to find out what resources are available, and ALPS then makes the reservation.

Currently ALPS is a "gang scheduler" and only allows one "job" per node. If a user submits a job in the format aprun -n 1 a.out , ALPS will put that job on one core of one node and leave the other seven cores empty. When the next job comes in, either from the same user or a different one, it will schedule that job to the next node.

If the user submits a job with aprun -n 10 a.out , then the scheduler will put the first eight tasks on the first node and the next two tasks on the second node, again leaving six empty cores on the second node. The user can modify the placement with -N , -S , and -cc .

A user might also run a single job with multiple treads, as with OpenMPI. If a user runs this job aprun -n 1 -d 8 a.out , the job will be scheduled to one node and have eight threads running, one on each core.

You can run multiple, different binaries at the same time on the same node, but only from one submission. Submitting a script like this will not work:

```
OMP_NUM_THREADS=1 aprun -n 1 -d 1 -cc 0 ./my-binary
OMP_NUM_THREADS=1 aprun -n 1 -d 1 -cc 1 ./my-binary
OMP_NUM_THREADS=1 aprun -n 1 -d 1 -cc 2 ./my-binary
OMP_NUM_THREADS=1 aprun -n 1 -d 1 -cc 3 ./my-binary
OMP_NUM_THREADS=1 aprun -n 1 -d 1 -cc 4 ./my-binary
OMP_NUM_THREADS=1 aprun -n 1 -d 1 -cc 5 ./my-binary
OMP_NUM_THREADS=1 aprun -n 1 -d 1 -cc 6 ./my-binary
OMP_NUM_THREADS=1 aprun -n 1 -d 1 -cc 7 ./my-binary
```

This will run a job on each core, but not at the same time. To run all jobs at the same time, you need to first add all the binaries within one aprun command:

```
$ cat run-all.pbs
./my-binary1
./my-binary2
./my-binary3
./my-binary4
./my-binary5
./my-binary6
./my-binary7
./my-binary8
$ aprun -n 1 run.pbs
```

Alternatively, use the command aprun -n 1 -d 8 run.pbs. To run multiple serial jobs, you must build a batch script to divide the number of jobs into groups of eight, and the

11.1.5 Interactive Queues

The current queing system contains the ability to run interactive queues. This is quite usefule, if you need to debug programs interactively that you will run than in a bacth queue. To use this feature we provide here a simple exaple on how to use a node on bravo.

Start an interactive shell with X11 forwarding on bravo you have to first login into india as the bravo queues are currently controlled on india:

```
ssh -X india
```

Than you need to start an interactive node:

```
qsub -I -q bravo -X
```

As xterm is currently not installed on bravo, you can test the X11 forwarding with:

```
firefox
```

The best way is to find your own resources and let us know which we shoudl add.

CHAPTER

TWELVE

HARDWARE

12.1 Hardware at Indiana University

In this section we describe the hardware that is available at Indiana University and allows you as part of course work or joint projects to gain easily access to them.

12.1.1 Compute Resources

The tables *Table: Compute Resources* and *Table: Compute Resource Details* show an overview of some imporatnt information about these clusters.

Table 12.1: Table: Compute Resources

Name	System Type	# Nodes	# CPUS	# Cores	TFLOPS	RAM (GB)	Storage (TB)	Site
india	IBM iDataplex	128	256	1024	11	3072	335	IU
xray	Cray XT5m	1	166	664	6	1328	5.4	IU
bravo	HP Proliant	16	32	128	1.7	3072	128	IU
delta	SuperMicro GPU Cluster	16	32	192		1333	144	IU
echo	SuperMicro ScaleMP Cluster	16	32	192	2	6144	192	IU

Table 12.2: Table: Compute Resource Details

Name	India	Echo	Bravo	Delta	Xray
Organization	Indiana University		Indiana University	Indiana University	Indiana University
Machine Type	Cluster	Cluster SclaeMP	Cluster	Cluster	Cluster
System Type	IBM iDataPlex dx 360 M2	Super-Micro	HP Proliant		Cray XT5m
CPU type	Intel Xeon X5550	Intel Xeon E5-2640	Intel Xeon E5620	Intel Xeon 5660	AMD Opteron 2378
Host Name	india	echo	bravo	delta	xray
CPU Speed	2.66GHz	2.50GHz	2.40GHz	2.80 GHz	2.4GHz
Number of CPUs	256		32	32	168
Number of nodes	128	12	16	16	1
RAM	24 GB DDR3 1333Mhz		192 GB DDR3 1333Mhz	192 GB DDR3 1333 Mhz	8 GB DDR2-800
Total RAM (GB)	3072		3072	3072	1344
Number of cores	1024	144	128		672
Operating System	Linux		Linux	Linux	Linux
Tflops	11		1.7		6
Disk Size (TB)	335	2.8		15	335
Hard Drives	3000 GB Internal 7200 RPM SATA Drive		6x2TB Internal 7200 RPM SATA Drive	Seagate Constellation 7.2 K RPM 64 MB Cache SATA 92GB	6 TB Internal Lustre Storage
Primary storage shared by all nodes	NFS		NFS	NFS	NFS
Storage details				RAID 9260-4i 1pt SAS2 512 MB SGL	
Connection configuration	Mellanox 4x DDR InfiniBand adapters		Mellanox 4x DDR InfiniBand adapters		Cray SeaStar Interconnect
Primary storage shared by all nodes					
CPUs (cores) per node				2	
Cores per CPU				6	
Total number of GPU cores				192	
GPU type				nVIDIA Tesla C2070	
Cores per GPU				448	
GPUs per node				2	
Batch system				Torque	

12.1.2 Networks

Resource Name	Network Devices		
IU Cray	Cray 2D Torus SeaStar		
IU iDataPlex	DDR IB	QLogic switch with Mellanox ConnectX adapters	Blade Network Technologies & Force10 Ethernet switches

Below is further information about networking:

Resource	Network Switch	Link
FutureGrid Core	Juniper EX8200	
India	Force10 C-150	Juniper/Dell EX series Force 10
Bravo	Force10 S60	force10-s60
Delta	Force10 S60	
Echo	Force10 S60	
Xray	Force10, C-150	Force10-c150
Node NICs	built-in (IBM iDataPlex DX360 M2) dual Intel 82575EB Gigabit Network Connection 10Gbps, Myricom Myri-10G Dual-Protocol NIC (available on login node)	

CHAPTER

THIRTEEN

DEVOPS

13.1 Historical and Functionality Perspective

- Make (1977)
- GNU autotools
- rpm/yum
- DevOps

    *cfengine * puppet * chef

13.2 Makefile

To manage large software programs it is often necessary to recompile them and to just focus on the peaces of code that are new. Thus software developers have early on focussed on building software components and libraries that can be separately compiled and integrated in the overall program executable through for example libraries.

Unfortunately, this comes also at a price that the management of such "assembled" software can be quite complex and involves the compilation of code ina particular order or the creation of artifacts during compile time.

To coordinate such execution *Makefiles* have been popular as they provide the ability to integrate a simple structure in the compile process, while detecting changes to source code that itself invoke actions as part of the make process.

The coordination of the process is specified in a *makefile* that contains targets that get invoked based on conditions such as that a source file has changed. The target has a body attached with it that will be executed when the precondition to the target is fulfilled.

Through a series of targets relatively sophisticated compile workflows can be specified and often the developer has to just call the command:

```
make
```

In addition make has also the ability to execute the programs in parallel while using the option *-j*. For large programs this can provide a significant speedup during the program assembly.

A sample Makefile looks as follows:

```
all: ring

ring: main.o message.o ring.o
        g++ main.o message.o ring.o -o ring

main.o: main.cpp
```

```
        g++ -c main.cpp

message.o: message.cpp
        g++ -c message.cpp

ring.o: ring.cpp
        g++ -c ring.cpp

clean:
        rm -rf *o ring
```

This example makefile creates a program with the name ring while integrating the *ring.c* and the *message.c* code into a single executable called *ring*.

On Unix systems one can find out more about make when saying:

```
man make
```

Much more detailed information is provided at

- http://www.gnu.org/software/make/manual/make.html

13.2.1 Practical Other Applications of Make

If you look at the process on how we create the documentation of this Web page, you will also see a number of Makefiles. Although we do not create c, we do create a web pages based on Sphinx translating rst files to html pages. This indicates the versatility of make.

13.2.2 Exercises

1. Write a c++ program that prints "Hello Cloud". Use a library cloud.o and create the program hello with a Makefile

13.3 Shell Scripts

Often shells are used to interact with computers n the command line. they provide convenient access to a number of commands and functionality that makes interactive experiments through a series of command possible. Shells are interpreters that provide a minimal environment to allow easy scripting of commands as part of schell scripts. Features that are of the used are aliasses, command definition, function, and clearly batch operations by listing commands sequentially in a shell script. Programming language features such as loops and conditions are also provided.

As shell scripts are executed as part of the OS no further install is necessary to run them.

However in contrast to modern programming languages and interpreters some functionality is missing. Also large amount of shell scripts become quite difficult to maintain due to the lack of more modern programming language features.

Therefor it is today common to use perl and more important;y python for the development of large scale scripts.

However, a large number of "tricks" and existing scripts makes shell scripts still a viable option for many developers. In addition it has the advantage that it will be immediately available after the install of the OS, thus it is of great help in case of managing distributed machines.

Variants of shells exist that can execute the same command in parallel on multiple distributed machines which comes in handy for cluster management.

13.3.1 Exercises:

1. Write a shell script that prints you username and lists the files in your home directory.
2. Write a shell script that converts all jpg files to png. (If you copy the example form http://en.wikipedia.org/wiki/Shell_script please make sure to walk through the example and understand in detail the syntax and the meaning of the program).
3. Search in our page for the term "pdsh"

13.4 Configure

Due to the many different computer systems it is important o note that libraries compiled on one system may not exist on another system or that different files need to be involved as part of the make process on the various systems. For example a command may not exist with the same name on the different computers.

To assit in this task developers use configure scripts that adapt to the underlaying enviorinment by abstarcting system related dependencies and fulfilling them with concrete implementations. As part of this process the Makefile will typically created and also a target install will be defined in the Makefile tu guide the insatlation process. Hence the process usually looks like:

```
./confugure
make
make install
```

A good image showcaseing all component involved in the configure process is available in Wikipedia. It depicts how a compatible Makefile can be generated. Developers will work on creating a Makefile.in with the available tools, that than will be used as the input to configure to create the actual makefile.

13.4.1 Excersises

1. What is autoconf and auomake?
2. Showcase the development of a Makefile.in for our Hello cloud example.
3. Use configure to deploy it

13.5 Package Managers

RPM

- http://en.wikipedia.org/wiki/RPM_Package_Manager

YUM

- http://en.wikipedia.org/wiki/Yellowdog_Updater,_Modified

apt-get

- https://help.ubuntu.com/community/AptGet/Howto

CHAPTER

FOURTEEN

IPYTHON

14.1 IPython

Notebook

IPython is a python command shell with notebook features that can be accessed through a browser. Throughout the material presented here we will be using IPython to present some of the demonstrations. This also allows you to try out the various excersises easily. We present here just a small set of features and recommend you to visit the IPython manaul for more information

14.1.1 Command Execution

To execute a shell command you can specify the ! at the beginning of a line

```
!echo "hallo"
```

```
hallo
```

14.1.2 Environment Variables

Environment variables can be accessed with $$

```
!echo "$$EDITOR"
```

```
emacs
```

Variables can be accessed by assigning values with = and by using them in { }

14.1.3 Variables

```
a="Hallo"
```

```
!echo "{a}"
```

```
Hallo
```

14.1.4 Surpressing output

Output can be surpressed while using the %%capture command. These commands are called magic functions in IPython. Many more magic functions are documented in the IPython manual

```
%%capture
!echo "You can not see me"

!echo "You can see me"
```

```
You can see me
```

CHAPTER

FIFTEEN

RESTRUCTUREDTEXT

Cheatcheat

- http://github.com/ralsina/rst-cheatsheet/raw/master/rst-cheatsheet.pdf
- http://docutils.sourceforge.net/docs/ref/rst/directives.html

Important extensions:

- http://sphinx-doc.org/ext/todo.html

15.1 Sections

RST allows to specify a number of sections. You can do this with the various underlines:

```
**********
Chapter
**********
Section
==========
Subsection
----------
Subsubsection
^^^^^^^^^^^^^^
Paragraph
~~~~~~~~~~~~~~
```

15.2 Listtable

```
.. csv-table:: Eye colors
   :header: "Name", "Firstname", "eyes"
   :widths: 20, 20, 10

   "von Laszewski", "Gregor", "gray"
```

Table 15.1: a title

Name	Firstname	eyes
von Laszewski	Gregor	gray

15.3 Exceltable

we have integrated Excel table from http://pythonhosted.org//sphinxcontrib-exceltable/ intou our sphinx allowing the definition of more elaborate tables specified in excel. Howere the most convenient way may be to use list-tables. The documentation to list tables can be found at http://docutils.sourceforge.net/docs/ref/rst/directives.html#list-table

15.4 Boxes

15.4.1 Seealso

```
.. seealso:: This is a simple **seealso** note.
```

See also:

This is a simple **seealso** note.

15.4.2 Note

Note: This is a **note** box.

```
.. note::  This is a **note** box.
```

15.4.3 Warning

Warning: note the space between the directive and the text

```
.. warning:: note the space between the directive and the text
```

15.4.4 Others

Attention: This is an **attention** box.

```
.. attention:: This is an **attention** box.
```

Caution: This is a **caution** box.

```
.. caution:: This is a **caution** box.
```

Danger: This is a **danger** box.

```
.. danger:: This is a **danger** box.
```

Error: This is a **error** box.

```
.. error:: This is a **error** box.
```

Hint: This is a **hint** box.

```
.. hint:: This is a **hint** box.
```

Important: This is an **important** box.

```
.. important:: This is an **important** box.
```

Tip: This is a **tip** box.

```
.. tip:: This is a **tip** box.
```

15.5 Sidebar directive

It is possible to create sidebar using the following code:

```
.. sidebar:: Sidebar Title
    :subtitle: Optional Sidebar Subtitle

    Subsequent indented lines comprise
    the body of the sidebar, and are
    interpreted as body elements.
```

Sidebar Title

Optional Sidebar Subtitle

Subsequent indented lines comprise the body of the sidebar, and are interpreted as body elements.

15.6 Autorun

Autorun is an extension for Sphinx_ that can execute the code from a runblock directive and attach the output of the execution to the document.

For example:

```
.. runblock:: pycon

    >>> for i in range(3):
    ...     print i
```

Produces

```
>>> for i in range(3):
...     print i
...
0
```

```
1
2
```

Another example:

```
.. runblock:: console

    $ date
```

Produces

```
$ date
Tue Sep 30 04:39:48 EDT 2014
```

However, when it comes to excersises we do preferthe use of ipython notebooks as this allows us to present them also to users as self contained excersises.

15.7 Hyperlinks

Direct links to html pages can ve done with:

```
`This is a link to an html page <hadoop.html>`_
```

Note that this page could be generated from an rst page

Links to the FG portal need to be formulated with the portal tag:

```
:portal:`List to FG projects </projects/all>`
```

In case a subsection has a link declared you can use :ref: (this is the prefered way as it can be used to point even to subsections:

```
:ref:`Connecting private network VMs  clusters <_s_vpn>`
```

A html link can be created anywhere in the document but must be unique. for example if you place:

```
.. _s_vpn:
```

in the text it will create a target to which the above link points when you click on it

15.8 Todo

```
.. todo:: an example
```

Todo

an example

CHAPTER

SIXTEEN

CREATE CLOUDMESH DEVELOPMENT IMAGE

16.1 How to setup a Cloudmesh Image

Note: The documentation talks about the steps to be followed to create a image with cloudmesh installed and boot a new VM from the cloudmesh image. The steps to be followed are:

```
- Boot a VM from an existing image using nova commands
- Setup Cloudmesh on the new VM
- Create an image from the new VM
- Test the image
```

16.1.1 Boot a VM from an existing image using nova commands

The steps to boot a VM from an existing image is available in the link: http://cloudmesh.github.io/introduction_to_cloud_computing/iaas/openstack.html. Follow the steps until "List running images".

For the sake of convenience, the steps without description is listed below:

```
$ ssh username@india.futuregrid.org
$ module load novaclient
$ source ~/.futuregrid/openstack_havana/novarc
$ nova flavor-list
$ nova image-list
$ nova keypair-add $USER-key > ~/.ssh/$USER-key
$ chmod 600 ~/.ssh/$USER-key
$ nova keypair-list
$ nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0
$ nova secgroup-add-rule default tcp 22 22 0.0.0.0/0
$ nova secgroup-list-rules default
$ nova boot --flavor m1.large \
            --image "futuregrid/ubuntu-14.04" \
            --key_name $USER-key $USER-001
$ nova list
```

If everything worked fine, the output of the last command should be similar to:

```
+--------------------------------------+------------+--------+------------+-------------+----------
| ID                                   | Name       | Status | Task State | Power State | Networks
+--------------------------------------+------------+--------+------------+-------------+----------
| 0747fe47-f8b1-4fd9-8f51-ad694708e6b7 | <USER>-001 | ACTIVE | None       | Running     | private=1
+--------------------------------------+------------+--------+------------+-------------+----------
```

16.1.2 Setup Cloudmesh on the new VM

Once the status becomes 'ACTIVE', log in to the VM using the IP address given in the 'Networks' column:

```
$ ssh -l ubuntu -i ~/.ssh/$USER-key 10.39.1.16
```

Note: This IP address is a private IP address and can be reached only when you are logged in to the 'India' machine.

Preparation of the VM

Update the operating system by executing the following command:

```
$ sudo apt-get update
```

Install virtualenv and git by executing the following commands:

```
$ sudo apt-get install python-virtualenv
```

Note: This might take a minute or two to install

Once you have installed virtualenv, execute the following command to create a virtual environment:

```
$ virtualenv ~/ENV
```

Activate the virtual environment by executing the followoing command:

```
$ source ~/ENV/bin/activate
```

Execute the following command to permenantly put the source step into the bashrc file:

```
$ echo "source ~/ENV/bin/activate" >> ~/.bashrc
```

Install git by executing the following command:

```
$ sudo apt-get install git
```

Download cloudmesh code from github by running the following commands:

```
$ cd ~
$ git clone https://github.com/cloudmesh/cloudmesh.git
```

The above steps would have created a directory called cloudmesh with the codeset. Execute the following commands to setup required packages:

```
$ cd cloudmesh
```

The next step is to fix some IP addresses on India by execting the command:

```
$ ./bin/fix-india-routing.sh
```

Install Cloudmesh

Install the required packages by exectuing the following commands:

```
$ sudo ./install system [Note: This step would take a while. So be patient]
$ ./install requirements [Note: This will take a minute or two to complete]
```

Create the yaml configuration files by executing the following command:

```
$ ./install new
```

Install the cloudmesh code into the virtual environment, by running the following command:

```
$ ./install cloudmesh
```

This step will install the cloudmesh related executable in "~/ENV/bin/" directory.

Now we are at a point where we have the cloudmesh code installed successfully. After this there are certain steps which are specific to each user.

So we stop here and snapshot the VM instance.

16.1.3 Create an image from the new VM

Exit from the VM, by typing "exit" at the command prompt.

To view the existing images, execute the following command:

```
$ nova image-list
```

To snapshot the VM instance, execute the following command:

```
$ nova image-create $USER-001 futuregrid/cloudmesh-ubuntu-14.04
```

When the image is being created, the Status will be set to "SAVING". To check the status of the image creation execute the following command:

```
$ nova image-list
```

You would see an output similar to:

```
+--------------------------------------+-----------------------------------+--------+-------------
| ID                                   | Name                              | Status | Server
+--------------------------------------+-----------------------------------+--------+-------------
| 18c437e5-d65e-418f-a739-9604cef8ab33 | futuregrid/fedora-18              | ACTIVE |
| 1a5fd55e-79b9-4dd5-ae9b-ea10ef3156e9 | futuregrid/ubuntu-14.04           | ACTIVE |
| 592e74c1-6d7f-4518-8a9e-c37e2145d92a | futuregrid/cloudmesh-ubuntu-14.04 | SAVING | d917ac67-1ae
+--------------------------------------+-----------------------------------+--------+-------------
```

Once the image has been created successfully, the status is changed to ACTIVE as shown below:

```
+--------------------------------------+-----------------------------------+--------+-------------
| ID                                   | Name                              | Status | Server
+--------------------------------------+-----------------------------------+--------+-------------
| 18c437e5-d65e-418f-a739-9604cef8ab33 | futuregrid/fedora-18              | ACTIVE |
| 1a5fd55e-79b9-4dd5-ae9b-ea10ef3156e9 | futuregrid/ubuntu-14.04           | ACTIVE |
| 592e74c1-6d7f-4518-8a9e-c37e2145d92a | futuregrid/cloudmesh-ubuntu-14.04 | ACTIVE | d917ac67-1ae
+--------------------------------------+-----------------------------------+--------+-------------
```

This snapshot image has now cloudmesh codeset installed.

16.1.4 Test the image

Once the cloudmesh image has been created, We will go ahead and create a new VM instance from this snapshot:

```
$ nova boot --flavor m1.large \
            --image "futuregrid/cloudmesh-ubuntu-14.04" \
            --key_name $USER-key $USER-cloudmesh-001
```

Check the status of the vm creation by executing the command:

```
$ nova list
```

You would see an output similar to:

```
+--------------------------------------+-----------------------+--------+------------+-------------
| ID                                   | Name                  | Status | Task State | Power State
+--------------------------------------+-----------------------+--------+------------+-------------
| 0747fe47-f8b1-4fd9-8f51-ad694708e6b7 | <USER>-cloudmesh-001  | ACTIVE | None       | Running
+--------------------------------------+-----------------------+--------+------------+-------------
```

The IP generated by default would be a private IP and this IP is not accessible from external network. Follow the steps below to generate an external IP address which allows the VM to be accessible from external network:

```
$ nova floating-ip-create
$ export MYIP==`nova floating-ip-list | fgrep "None" | cut -d '|' -f2 | head -1`
$ nova add-floating-ip $USER-cloudmesh-001 $MYIP
```

Login to the VM by running the following command:

```
$ ssh -l ubuntu -i $USER-key  $MYIP
```

Once you are inside the VM, check if a cloudmesh directory is available by executing the command:

```
$ ls ~/cloudmesh
```

Update the user profile, name and project data in the cloudmesh.yaml file:

```
$ vi ~/.cloudmesh.yaml
```

An alternative way to do update the cloudmesh.yaml file is use the functionality provided by cloudmesh:

```
$ export PORTALNAME=<put your portal name here>
$ ssh-keygen -t rsa -C $PORTALNAME-ubuntu-vm-key
```

Now add the key to the ssh agent:

```
$ eval `ssh-agent -s`
$ ssh-add
```

The ssh key for the VM needs to be added to the FutureSystems portal account. Copy the ssh key from the file:

```
$ cat ~/.ssh/id_rsa.pub
```

Now fetch the user information needed to access openstack from India by running the following commands:

```
$ cm-iu user fetch
$ cm-iu user create
```

Note: When this command is run, it would ask for a portal username. Enter your FutureSystems portal name.

Make some changes to the India OpenStack configuration by executing the following command:

```
$ fab india.configure
```

Create the cloudmesh database by executing the following command:

```
$ fab mongo.reset
```

Start the cloudmesh services by executing the following command:

```
$ fab server.start
```

Access Cloudmesh using the http link:

```
http://PUBLIC_IP_OF_THE_VM:5000
```

Note: You can get the public ip of the VM by running the command:

```
$ echo $MYIP
```

Attention: Once you are done with your work and you no longer need the VMs, you can delete them with the commands:

```
$ nova delete $USER-cloudmesh-001
$ nova delete $USER-001
```

CHAPTER

SEVENTEEN

TODOS

Todo

Hyungro, check with allan and Koji if we have pdsh on india. At this time we are not aware that pdsh is installed by default on india. check with the systems group and have them provide a documentation on how we activate it.

(The *original entry* is located in /Users/flat/github/introduction_to_cloud_computing/docs/source/parallel.rst, line 48.)

Todo

an example

(The *original entry* is located in /Users/flat/github/introduction_to_cloud_computing/docs/source/rst.rst, line 236.)

Todo

Hyungro, update this example so it fits better in 80 column,

(The *original entry* is located in /Users/flat/github/introduction_to_cloud_computing/docs/source/cloudmesh/cm/.ipynb_checkpoints/_cr
vm-management-checkpoint.rst, line 189.)

Todo

Hyungro, add an example where you first set the default cloud

(The *original entry* is located in /Users/flat/github/introduction_to_cloud_computing/docs/source/cloudmesh/cm/.ipynb_checkpoints/_cr
vm-management-checkpoint.rst, line 399.)

Todo

Hyungro, update this example so it fits better in 80 column,

(The *original entry* is located in /Users/flat/github/introduction_to_cloud_computing/docs/source/cloudmesh/cm/.ipynb_checkpoints/_vr
cm-checkpoint.rst, line 277.)

Todo

get the cloud credentials form a yaml file

(The *original entry* is located in /Users/flat/github/introduction_to_cloud_computing/docs/source/proposed/_vm_sh.rst, line 17.)

Todo

use the information to start a VM

(The *original entry* is located in /Users/flat/github/introduction_to_cloud_computing/docs/source/proposed/_vm_sh.rst, line 19.)

Todo

do this section later as we need flavors and images

(The *original entry* is located in /Users/flat/github/introduction_to_cloud_computing/docs/source/proposed/_vm_sh.rst, line 21.)

CHAPTER

EIGHTEEN

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*
- *notebooks*