Introduction to Cloud Computing

Release Learning

Editor: Gregor von Laszewski

September 19, 2014

PREFACE

1.1 Conventions

test

$ 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 **portalname**.

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

Menu selections: *Start → Programs*

Man page: *ls(1)*

1.2 Using the Notebooks

The material provided in this documentation contains a number of IPython notebooks. These notebooks can be executed or accessed through te 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.

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
```

Next you have to compile the information with:

```
fab doc.html
```

Now you can start the notebook server with:

```
fab doc.notebook
```

The lit of notebooks will be provided in the sidebar menu. You can ow visit them and execute them. Howler, we like to remind you that this requires that you have previously setup cloudmesh.

To view the information locally you can say:

```
fab doc.html
```

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)

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

CLOUD ACCOUNTS

4.1 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

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 *s-account-create-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 *s-account-openid*.
4wjVwQbOlSI	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 *s-portal-sshkey*.
DzbLS6iCeTk	6:25 min	Create a project	This lesson explains you how to create a FutureSystems project. For written material, see section *s-account-create-account*.
5xQiPBwt58s	1:28 min	Join a project	This lesson explains you how to join a FutureSystems project. For written material, see section *s-account-join-project*.

4.1.1 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 *s-account-join-project*.

4. Read up on what a ssh key is.

   See section *Using SSH Keys*.

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

4.2 Using SSH Keys

Page Contents

- Using SSH from Windows
- Generate a SSH key
- Add or Replace Passphrase for an Already Generated Key
- Upload the key to the FutureSystem Portal
- Testing your ssh key

Hint:
If you do not know what ssh is we recommend that you read up on it . However, the simple material presented here will help you getting started quickly on FutureSystem. Some screencasts about this topic are available in section *Creating FutureSystems Accounts and Projects*.

To access the various FutureSystem resources, you need to provide a public ssh key to FutureSystem. We explain how to generate a ssh key, upload it to the FutureSystem portal and log onto the resources. This manual covers both UNIX and Windows Users.

4.2.1 Using SSH from Windows
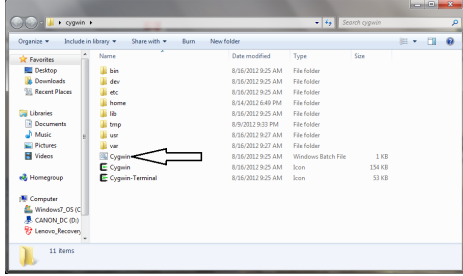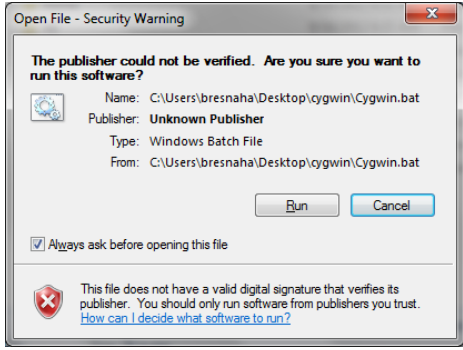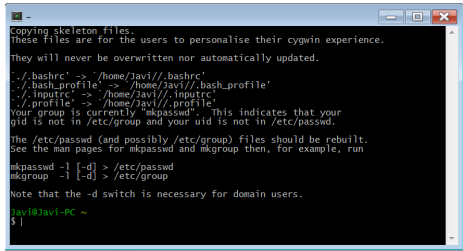
Hint: For Linux users, please skip to the section *Generate a SSH key*

Windows users need to have some special software to be able to use the SSH commands. If you have one that you are comfortable with and know how to setup key pairs and access the contents of your public key, please feel free to use it. The most popular software making ssh clients available to Windows users include

- cygwin

- putty

- or installing a virtualiztion software and running Linux virtual machine on your Windows OS.

For a real quick start we recommend you use Cygwin (Linux-like environment for Windows) because it will ease your experience with FutureSystem and provide you with a command shell that is Linux like. If you have cygwin already installed, please use it, but make sure you have ssh installed. If not, we have made it even easier for you as we prepared a special Cygwin version that is ready to use. Once you have installed it, you can follow the same instructions as given in the rest of the sections presented to access FutureSystem from ssh. You can install cygwin it with the following simple steps.

Step	Description	Supporting Screenshot
Step 1	Download Cygwin from our Portal https://portal.futuregrid.org/sites/default/files/cygwin.zip.	
Step 2	Uncompress the file.	
Step 3	Execute the file the 'Windows Batch File' called Cygwin.bat	
Step 4	You may get a warning. Click on the Run button	
Step 5	You get a Linux-like terminal that will allow you to continue with this manual. Hint: When showing examples of commands, the $ symbol precedes the actual command. So, the other lines are the output obtained after executing the command.	

4.2.2 Generate a SSH key

> **ℹ Hint**
>
> In case you do not want to type in your password everytime, please learn about ssh-agent and ssh-add.

First we must generate a ssh key with the tool ssh-keygen. This program is commonly available on most UNIX systems (this includes Cygwin if you installed the ssh module or use our pre-generated cygwin executable). It will ask you for the location and name of the new key. It will also ask you for a passphrase, which you **MUST** provide. Some teachers and teaching assistants advice you to not use passphrases. This is **WRONG** as it allows someone that gains access to

your computer to also gain access to all resources that have the public key. Also, please use a strong passphrase to protect it appropriately.

In case you already have a ssh key in your machine, you can reuse it and skip this whole section.

To generate the key, please type:

Example:

```
ssh-keygen -t rsa -C localname@indiana.edu
```

This command requires the interaction of the user. The first question is:

```
Enter file in which to save the key (/home/localname/.ssh/id_rsa):
```

We recommend using the default location ~/.ssh/ and the default name id_rsa. To do so, just press the enter key.

> ### Hint
>
> Please note that your *localname* is the username on your computer and may be different from your *portalusername*.

The second and third question is to protect your ssh key with a passphrase. This passphrase will protect your key because you need to type it when you want to use it. Thus, you can either type a passphrase or press enter to leave it without passphrase. To avoid security problems, you **MUST** chose a passphrase. Make sure to not just type return for an empty passphrase:

```
Enter passphrase (empty for no passphrase):
```

and:

```
Enter same passphrase again:
```

If executed correctly, you will see some output similar to:

```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/localname/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/localname/.ssh/id_rsa.
Your public key has been saved in /home/localname/.ssh/id_rsa.pub.
The key fingerprint is:
34:87:67:ea:c2:49:ee:c2:81:d2:10:84:b1:3e:05:59 localname@indiana.edu
The key's random art image is::

+--[ RSA 2048]----+
|.+...Eo= .       |
| ..=.o + o +o    |
|O.  o o +.o      |
| = .   . .       |
+-----------------+
```

Once, you have generated your key, you should have them in the .ssh directory. You can check it by

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

If everything is normal, you will see something like:

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCXJH2iG2FMHqC6T/U7uB8kt6KlRh4kUOjgw9sc4Uu+Uwe/EwD0wk6CBQMB+HKb
```

4.2.3 Add or Replace Passphrase for an Already Generated Key

In case you need to change your change passphrase, you can simply run "ssh-keygen -p" command. Then specify the location of your current key, and input (old and) new passphrases. There is no need to re-generate keys:
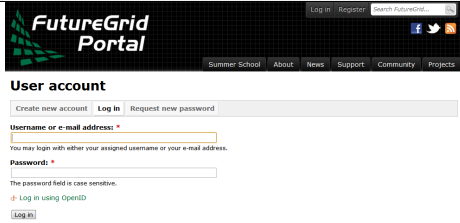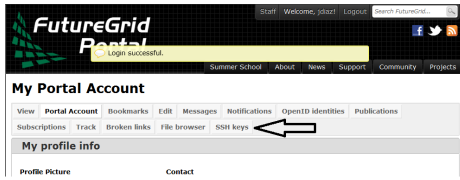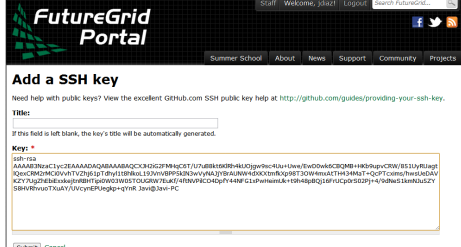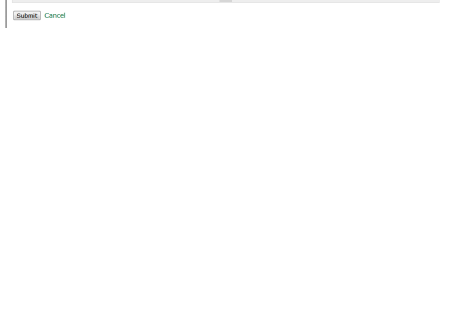
```
ssh-keygen -p
```

You will see the following output once you have completed that step:

```
Enter file in which the key is (/home/localname/.ssh/id_rsa):
Enter old passphrase:
Key has comment '/home/localname/.ssh/id_rsa'
Enter new passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved with the new passphrase.
```

4.2.4 Upload the key to the FutureSystem Portal

Next you need to upload the key to the portal. You must be logged into the portal to do so.

Step Description	Supporting Screensho
Step 1	Log into the portal	
Step 2	Click in the "ssh key" button. or go directly to https://portal.futuregrid.org/my/ssh-keys	
Step 3	Click in the "add a public key" link.	
Step 4	Paste your ssh key into the box marked Key. Use a text editor to open the "id_rsa.pub". Copy the entire contents of this file into the ssh key field as part of your profile information. Many errors are introduced by users in this step as they do not paste and copy correctly.	
Step 5	Click the submit button. **IMPORTANT**: Leave the Title field blank. Make sure that when you paste your key, it does not contain newlines or carriage returns that may have been introduced by incorrect pasting and copying. If so, please remove them.	

At this point you have uploaded your key. However you will still need to wait till all accounts have been set up to use the key, or if you did not have an account till it has been created by an administrator. Please, check your email for further updates. You can also refresh this page and see if the boxes in your account status information are all green. Than you can continue.

4.2.5 Testing your ssh key

If you have had no FutureSystem account before, you need to wait for up to two business days so we can verify your identity and create the account. So please wait. Otherwise, testing your new key is almost instantaneous on india. For

other clusters like it can take around 30 minutes to update the ssh keys.

To log into india simply type the usual ssh command such as:

```
$ ssh portalname@india.futuregrid.org
```

The first time you ssh into a machine you will see a message like this:

```
The authenticity of host 'india.futuregrid.org (192.165.148.5)' can't be established.
RSA key fingerprint is 11:96:de:b7:21:eb:64:92:ab:de:e0:79:f3:fb:86:dd.
Are you sure you want to continue connecting (yes/no)? yes
```

You have to type yes and press enter. Then you will be logging into india. Other FutureSystem machines can be reached in the same fashion. Just replace the name india, with the appropriate FutureSystems resource name.

FIVE

IPYTHON

5.1 IPython

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

5.1.1 Command Execution

To execute a shell command you can specify the ! at the beginning of a line

```
!echo "hallo"
```

```
hallo
```

5.1.2 Environment Variables

Environment variables can be accessed with $$

```
!echo "$$EDITOR"
```

```
emacs
```

Variables can be accessed by assigning values with = and by using them in { }

5.1.3 Variables

```
a="Hallo"
```

```
!echo "{a}"
```

```
Hallo
```

5.1.4 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
```

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

6.1 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.

6.2 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:

```python
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
```

6.3 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.

6.3.1 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

6.3.2 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.

```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
```

```python
# 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.

6.4 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.

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 FutureGrid infrastructure (4704 cores used by more than 355 projects), 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.

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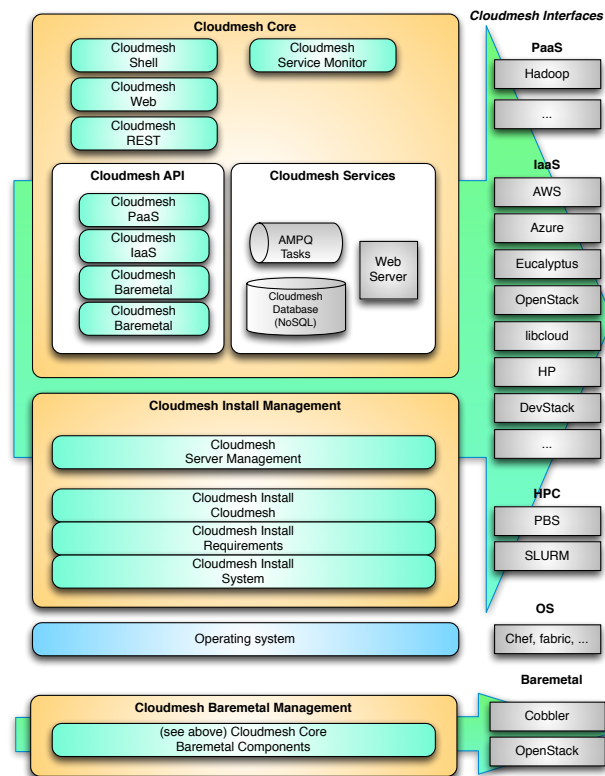
Figure 7.2: **Figure:** *High level architecture of Cloudmesh*

- 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 FutureGrid.

7.1.6 Cloudmesh HPC

Cloudmesh provides an easy tou use API and GUI to HPC queues. It allows simple display of queues. As FutureGrid 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.

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.

7.1.10 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.

7.1.11 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.

7.1.12 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 Quick Start on your desktop

> **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.

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 [username]@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 portal name 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 [project id]
```

```
*[project id]* 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.

First, please set the PORTALNAME in your shell:

```
export PORTALNAME=[name of your user account in FutureSystem]
```

For ubuntu use

```
git clone https://github.com/cloudmesh/cloudmesh.git
virtualenv ~/ENV
source ~/ENV/bin/activate
cd cloudmesh
sudo ./install system
```

Next execute:

```
./install requirements
./install new
./install cloudmesh
cm-iu user fetch --username=$PORTALNAME
cm-iu user create
fab mongo.reset
```

Next execute:

```
fab server.start
cm cloud list
cm cloud on india
cm flavor india --refresh
```

For OSX use

```
git clone https://github.com/cloudmesh/cloudmesh.git
virtualenv ~/ENV
source ~/ENV/bin/activate
cd cloudmesh
./install system
```

Next execute:

```
./install requirements
./install new
./install cloudmesh
cm-iu user fetch --username=$PORTALNAME
cm-iu user create
fab mongo.reset
```

Next execute:

```
fab server.start
cm cloud list
cm cloud on india
cm flavor india --refresh
```

One line install with curl

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=[your FutureSystems portal id] 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 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 USERNAME@india.futuregrid.org hostname
ssh USERNAME@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: flat]:
```

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.

7.2.3 Quickstart for an Openstack VM

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. This is described in the following document:

More information about horizon on FutureSystems is available at our manual page.

We summarize the following steps, however like to point out that the best source for this is our previously pointed out document:

```
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-key
```

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

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
```

And then 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 $USER

india$ nova floating-ip-create

india$ export MYIP=`nova floating-ip-list |fgrep "None" | cut -d '|' -f2`
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                              | hsakjfhsdlkjfhsdlkjhflskjdhflkjsdhflkjshfpoeuiyrewuohfkljsdl
| 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. 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:

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

This will fetch your cloud credentials from FutureSystems and populate them into the yaml config file. BEFORE you can do this, make sure you can log into the FutureSystems resources, e.g. india. You will need a private key present in the VM that the matching public ssh key had been registered to the FutureSystems. Additionally you may need to excetue the following beforehand to add your password protected key into the session:

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

To run cloudmesh you will need to start a number of services that you can do with:

```
fab mongo.boot
```

In some cases you may see connection problems in the later step. In that case please execute this command one again so the tables and security settings are done properly.

Once the mongo is initiated properly it's time to update the user data with:

```
fab user.mongo
```

Before you start the server, you need to execute this so the server would be accessible from outside:

```
fab india.configure
```

And then start the server:

```
fab server.start
```

Then the cloudmesh service should be available via:

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

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

7.2.4 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
```

7.2.5 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 use and launch 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

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

7.3 Cloudmesh cm

7.3.1 Cloudmesh `cm` Command

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
```

```
1.0.4
```

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
```

```
1.0.4
```

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       edit     help       label   plugins  register        user
banner    exec     image      list    project  script          var
clear     exp      info       man     py       security_group  verbose
cloud     flavor   init       metric  q        storm           version
defaults  graphviz inventory  open    quit     timer           vm
dot2      group    keys       pause   rain     use             web

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

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

Starting the Web Browser

To start the browser, simply type the command

Listing Clouds

```
cm cloud list
```

```
+---------+----------+
| cloud   | active   |
+=========+==========+
| alamo   |          |
+---------+----------+
| aws     |          |
+---------+----------+
| azure   |          |
+---------+----------+
| hp      |          |
+---------+----------+
| hp_east |          |
+---------+----------+
| india   | True     |
+---------+----------+
| sierra  | 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]
        cloud info [CLOUD|--all]
        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      |
+=========+==========+============+
| alamo   |          | alamo      |
+---------+----------+------------+
| aws     |          | aws        |
+---------+----------+------------+
| azure   |          | waz        |
+---------+----------+------------+
| hp      |          | hpos       |
+---------+----------+------------+
| hp_east |          | hpeos      |
+---------+----------+------------+
| india   | True     | ios_havana |
+---------+----------+------------+
```

```
| sierra  | True     | sos        |
+---------+----------+------------+
```

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`

The `project` command provides a list of default, active, or completed project fo 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: fg20
  projects: fg20, fg82
 completed: fg130
```

Set a default project

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

```
cm project default fg82

CM ... site/cloudmesh-1.0-py2.7.egg/cloudmesh_cmd3/plugins/cm_shell_project.pyc:58:   INFO - sets the
CM ... site/cloudmesh-1.0-py2.7.egg/cloudmesh_cmd3/plugins/cm_shell_project.pyc:70:   INFO - fg82 pro
fg82 project is a default project now
```

```
cm project

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

   default: fg82
  projects: fg20, fg82
 completed: fg130
```

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

CM ... site/cloudmesh-1.0-py2.7.egg/cloudmesh_cmd3/plugins/cm_shell_project.pyc:75:   INFO - Sets the
CM ... site/cloudmesh-1.0-py2.7.egg/cloudmesh_cmd3/plugins/cm_shell_project.pyc:83:   INFO - fg415 pr
fg415 project is an active project(s) now

cm project info

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

   default: fg82
  projects: fg20, fg82, fg415
 completed: fg130
```

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

CM ... site/cloudmesh-1.0-py2.7.egg/cloudmesh_cmd3/plugins/cm_shell_project.pyc:115:   INFO - Sets a
CM ... site/cloudmesh-1.0-py2.7.egg/cloudmesh_cmd3/plugins/cm_shell_project.pyc:126:   INFO - fg415 p
fg415 project is in a completed project(s)

cm project

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

   default: fg82
  projects: fg20, fg82
 completed: fg130, fg415
```

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
```

Future Extensions

Todo

Hyungro, please proovide an extension to this function to

allow adding and deleting of projects.

```
cm project add fg455
          cm project delete fg455
```

Adding and deletion of projects may need to be done carefully. We do not want t o add projects you are not member of. It would be important to add some test if possible. The –force flag forces the addition and deletion without test.

Todo

Hyungro, please add the –format=FORMAT flag to print the

output in json and a table.

7.3.3 `debug` Command in Cloudmesh `cm`

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
```

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

`debug on/off` Command

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
```

```
Debug mode is on.
```

```
cm debug off
```

```
CM ... site/cloudmesh-1.0-py2.7.egg/cloudmesh_cmd3/plugins/cm_shell_yaml.pyc:38:   INFO - Debug mode
Debug mode is off.
```

7.3.4 `loglevel` Command in Cloudmesh cm

`loglevel` allows you to update a log level on Cloudmesh. * `DEBUG` * `INFO` * `WARNING` * `ERROR` * `CRITICAL`

View a current log level

```
cm loglevel
```

```
ERROR
```

Set a log level between debug, info, warning, error, or critical.

```
cm loglevel debug
```

```
DEBUG mode is set.
```

7.3.5 `yaml` Command in Cloudmesh cm

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
```

`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
```

```
DEBUG
```

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

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
```

```
ERROR
```

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

7.3.6 `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 ... site/cloudmesh-1.0-py2.7.egg/cloudmesh_cmd3/plugins/cm_shell_yaml.pyc:56:  INFO - color on.
color on.
```

```
cm color off
```

```
CM ... site/cloudmesh-1.0-py2.7.egg/cloudmesh_cmd3/plugins/cm_shell_yaml.pyc:63:  INFO - color off.
color off.
```

7.3.7 CM Console for VM Management

Initialization (Preparing on IPython)

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 line

```
alias cm cm %s
```

```
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.

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 list
```

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

```
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"
```

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 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
```

```
+------+-----------+---------+-------+--------+----------------------+
|  id  | name      |  vcpus  |   ram |   disk | refresh time         |
+======+===========+=========+=======+========+======================+
|    1 | m1.tiny   |       1 |   512 |      0 | 2014-09-15T01-09-56Z |
+------+-----------+---------+-------+--------+----------------------+
|    3 | m1.medium |       2 |  4096 |     40 | 2014-09-15T01-09-56Z |
+------+-----------+---------+-------+--------+----------------------+
|    2 | m1.small  |       1 |  2048 |     20 | 2014-09-15T01-09-56Z |
+------+-----------+---------+-------+--------+----------------------+
|    5 | m1.xlarge |       8 | 16384 |    160 | 2014-09-15T01-09-56Z |
+------+-----------+---------+-------+--------+----------------------+
|    4 | m1.large  |       4 |  8192 |     80 | 2014-09-15T01-09-56Z |
+------+-----------+---------+-------+--------+----------------------+
```

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.

Todo

Hyungro, update this example so it fits better in 80 column,

---

maybe remove the id

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

```
+-----------------------------------------------------+------------------------------------+--------
| name                                                | id                                 | updated
+=====================================================+====================================+========
| salsahpc/cloud-mooc-m1-large-4GB                    | 384ca88c-f674-4d3d-999f-353fc6915608 | 2013-12
+-----------------------------------------------------+------------------------------------+--------
| CentOS6                                             | ad35042c-d242-4514-bde0-138718b5aa3e | 2014-04
+-----------------------------------------------------+------------------------------------+--------
| SL64-blank-sparse10gb-C vmdk                        | f480f1e7-38af-4410-bc13-6a1e43bf58ba | 2014-02
+-----------------------------------------------------+------------------------------------+--------
| fg7/rynge/centos6-v1                                | ac464f65-7175-44df-9e0e-34b1a32f8a2a | 2014-05
+-----------------------------------------------------+------------------------------------+--------
| futuregrid/centos-6                                 | 81b27cb5-4f8b-4583-afd4-1901053f6a28 | 2014-05
+-----------------------------------------------------+------------------------------------+--------
| DaLiAna-vm2014e-geant4.10.vmdk Jan best             | f4f9821f-881b-433a-97ab-1ad87273c5fc | 2014-03
+-----------------------------------------------------+------------------------------------+--------
| ubuntu-13.10                                        | cd844b12-f138-414a-8cd5-2c977f0a2379 | 2014-01
+-----------------------------------------------------+------------------------------------+--------
| futuregrid/ubuntu-14.04                             | ba327564-5969-4309-b3f3-b67764038e66 | 2014-09
+-----------------------------------------------------+------------------------------------+--------
| fg101/richieriee/my-ubuntu-01                       | 7915f335-d1a9-4380-a9f3-159eb67d721e | 2014-07
+-----------------------------------------------------+------------------------------------+--------
| sl64-gluex-vm2014e-40gb.vmdk Justin                 | f51fc1e4-d495-4376-824b-19000c41abef | 2014-03
+-----------------------------------------------------+------------------------------------+--------
| futuregrid/fedora-19                                | 6e9322df-fcb2-4ac8-bd02-1d39c6f6919f | 2014-09
+-----------------------------------------------------+------------------------------------+--------
| fg10/jcharcal/centos6.5_x86_64                      | 28f2ad75-8f42-4079-85e1-3d4fe986317f | 2014-03
```

```
+------------------------------------------------+--------------------------------------+--------
| balewski/kernel-2.6.32-431.5.1-sl65            | 00e935c3-82a6-499c-9056-6db37d27439b | 2014-0
+------------------------------------------------+--------------------------------------+--------
| snapshot 1                                     | adc6b8d0-8fba-47ba-af20-f91cb84334cc | 2014-08
+------------------------------------------------+--------------------------------------+--------
| Ubuntu-12.04-blank2.vmdk                       | b49324f8-d2c3-413a-8f8c-83b498e815f6 | 2014-02
+------------------------------------------------+--------------------------------------+--------
| balewski/ramdisk-2.6.32-431.5.1-sl65           | 9589cf93-f0e6-45d8-930f-63dd2d9c2f1a | 2014-03
+------------------------------------------------+--------------------------------------+--------
| sl6_x64-qemu  french ?bad                      | c9ca9852-dfcb-4ceb-a164-a58c89594551 | 2014-02
+------------------------------------------------+--------------------------------------+--------
| balewski/sl6.5-blank-80gb-b works              | 5cb90f55-c3a1-468b-b60a-0ecd589baf31 | 2014-03
+------------------------------------------------+--------------------------------------+--------
| futuregrid/ubuntu-12.04                        | 9cd8cc0d-96cd-44b8-a307-8b574c899ef4 | 2014-09
+------------------------------------------------+--------------------------------------+--------
| ndssl-vt/ubuntu-12.04-small                    | b9455041-407b-488b-a3f1-5dce6c480b5a | 2013-12
+------------------------------------------------+--------------------------------------+--------
| futuregrid/fedora-20                           | 68a5d085-ea4d-42c4-800f-2017309989ae | 2014-09
+------------------------------------------------+--------------------------------------+--------
| SL64-blank-sparse40GB vmdk                     | 882871a5-3157-426a-b7bc-10cca91860f0 | 2014-02
+------------------------------------------------+--------------------------------------+--------
| cglmoocs/ipython                               | 0eff8bfc-e455-429e-9dfb-e16488d410f9 | 2014-01
+------------------------------------------------+--------------------------------------+--------
| balewski/daliana-vm2014e2-sl6.5-geant4.10-root5.34 | 64866d96-8a9e-4d7c-983f-90b0bb594310 | 2014-03
+------------------------------------------------+--------------------------------------+--------
| DaLiAna-vm2014d-SL64-A.vmdk                    | ff5bb4de-d7e9-4bfb-b42b-cec9ed8d2d08 | 2014-02
+------------------------------------------------+--------------------------------------+--------
```

Quick Start a VM

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

```
cm "vm start --cloud=india"
```

Delete a VM

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

```
cm "vm delete [NAME] --cloud=india"
```

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"
```

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

prefix:  brian
index:  32
next VM name: brian_32
```

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

```
cm label

prefix:  Alex
index:  40
next VM name: Alex_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

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 5 VMs on india

```
cm "vm start --cloud=india --count=5"
```

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:

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 Brain_2 --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=abc --range=,100 --force"
```

For more details for command vm

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

Todo

Hyungro, add an example where you first set the default cloud

and than use `vm delete Brain_2`

7.3.8 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

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

```python
import cloudmesh
```

Version

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

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

```
1.0
```

7.4.2 Defining Names

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.

```python
import uuid
```

Machine dependent uuid

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

```python
uuid.uuid1()
```

```
UUID('f040aaf5-3242-11e4-bd35-600308a5f9d2')
```

Random Uuid

uuid4 creates random uuids.

```
uuid.uuid4()
```

```
UUID('ac332de1-faef-4054-aae8-6f32eb15f982')
```

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('e529ec9c-e40d-4e3b-b4db-4b46802bd4f6')
```

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

```
64005057163216856052777834181321712834L
```

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()
```

```
f04729ee324211e49889600308a5f9d2
```

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")
```

```
'gregorf0487de3324211e4b177600308a5f9d2'
```

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()
```

7.4.3 Load Command

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()
```

```
['alamo', 'hp_east', 'sierra', 'aws', 'hp', 'india', 'azure']
```

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
```

```
'Gregor'
```

```
user.lastname
```

```
'von Laszewski'
```

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")
```

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

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

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

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

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

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

```python
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.

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

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

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

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

```
['launcher']
```

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

```
['recipies']
```

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

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

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

```
{`description': `Deploys a Slurm cluster. One of the Vms is the Master, while the others re
 `image': `/static/img/launcher/mooc.png',
 `label': `slurm',
 `name': `Mooc',
 `parameters': [{`nodes': {`type': `integer', `value': 1}},
                {`other': {`type': `float', `value': 5.0}},
                {`types': {`type': `string',
                           `value': `enter vm or bare metal'}},
                {`selector': {`type': `selector',
```

```
                            `value': ['vm', `baremetal', `xyz']}},
                 {`cloud': {`type': `selector', `value': ['india', `sierra']}}],
 `script': {`type': `sh',
             `value': `echo ``hello'' {{ selector }} -np {{ nodes }}'}}
```

7.4.4 Batch Queues

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

```python
import cloudmesh

username = cloudmesh.load().username()

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

QStat

```python
qstat = india.qstat()
```

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

```python
qstat.keys()

['echo.futuregrid.org',
 'india.futuregrid.org',
 'delta.futuregrid.org',
 'bravo.futuregrid.org']

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

[u'1906163.i136',
 u'1906085.i136',
 u'1906155.i136',
 u'1906160.i136',
 u'1906164.i136',
 u'1906161.i136',
 u'1906086.i136',
 u'1906088.i136',
 u'1906084.i136',
 u'1906162.i136',
 u'1906087.i136',
 u'1906154.i136']

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

1906163.i136 H
1906085.i136 R
1906155.i136 R
```

```
1906160.i136 H
1906164.i136 H
1906161.i136 H
1906086.i136 R
1906088.i136 R
1906084.i136 R
1906162.i136 H
1906087.i136 R
1906154.i136 Q
```

QInfo

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

To list the queue names, simply print the keys.

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

```
['reserved', 'systest', 'batch', 'b534', 'long', 'interactive']
```

More information is available when you inspect the dict.

7.4.5 Cloudmesh API for VM Management

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
```

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()
```

Activate the user account

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

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

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="india", cm_user_id="Taylor", prefix="hoho", index=5, flavor="m1.small", i
```

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.

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. 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??*

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

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

Get Flavors or Images

Available flavors can be listed with the following function.

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

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

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

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

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

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')
```

Quick Start a VM

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

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

Delete VM

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.

```
server = result['server']['id']
mesh.delete(cloud, server, username)
```

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) * `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": "brian"}

```
result = mesh.start("india", "Taylor", index=5, prefix="hoho")
```

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'Alex_37'
```

`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.

```
mesh.vmname("Brian", 10)
```

```
u'Brian_10'
```

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

```
mesh.vmname_next()
```

```
u'Brian_11'
```

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

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

```
u'Alex_21'
```

Assign a public IP address to the VM

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

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

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.

```
mesh.wait(ipaddr=ip, command="uname -a", interval=5, retry=10)
```

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

Refreshing VM status

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

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

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

7.5 Cloudmesh Shell

7.5.1 Cloudmesh Shell for VM Management

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      edit     help     label   plugins  register        user
banner   exec     image    list    project  script          var
clear    exp      info     man     py       security_group  verbose
cloud    flavor   init     metric  q        storm           version
```

```
defaults   graphviz   inventory   open     quit     timer          vm
dot2       group      keys        pause    rain     use            web


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


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

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   |
+=========+==========+
| alamo   |          |
+---------+----------+
| aws     |          |
+---------+----------+
| azure   |          |
+---------+----------+
| hp      |          |
+---------+----------+
| hp_east |          |
+---------+----------+
| india   | True     |
+---------+----------+
| sierra  |          |
+---------+----------+
```

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")
```

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   |
+=========+==========+
| alamo   |          |
```

```
+---------+----------+
| aws     |          |
+---------+----------+
| azure   |          |
+---------+----------+
| hp      |          |
+---------+----------+
| hp_east |          |
+---------+----------+
| india   | True     |
+---------+----------+
| sierra  |          |
+---------+----------+
```

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")
```

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")
```

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

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

```
print cloudmesh.shell("vm start --cloud=india")
```

Delete a VM

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

7.5. Cloudmesh Shell 65

```
cloudmesh.shell("vm delete [NAME] --cloud=india")
```

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")
```

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")

prefix:  Brian
index:  32
next VM name: Brian_32

cloudmesh.shell("label --prefix=Alex --id=40")

cloudmesh.shell("label")

prefix:  Alex
index:  40
next VM name: Alex_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

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 5 VMs on india

```
print cloudmesh.shell("vm start --cloud=india --count=5")
```

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:

to delete all VMs of cloud india

```
print 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)

```
print cloudmesh.shell("vm delete --cloud=india abc_2 --force")
```

to delete a VM by group

```
print 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

```
print cloudmesh.shell("vm delete --cloud=india --group=testgroup --prefix=abc --range=,100 --force")
```

For more details for command vm

```
print cloudmesh.shell("vm -h")
```

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.

```python
import cloudmesh
```

```
cloudmesh.shell('list')
```

```
Error: Wrong Format
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 co
        '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.

```python
print cloudmesh.shell("list vm india --column=status,name,address,flavor,id,created")
```

```
+----------+----------+--------------------------------------+----------+----------------------+
| name     | status   | id                                   | flavor   | created              |
+==========+==========+======================================+==========+======================+
| Brain_8  | ACTIVE   | 9a447694-b52d-4f78-b489-ff8d7de694cc | m1.small | 2014-09-16T18:20:02Z |
+----------+----------+--------------------------------------+----------+----------------------+
| Brain_10 | ACTIVE   | 9208543e-cf9f-4f31-997b-40ea4443a808 | m1.small | 2014-09-16T18:20:04Z |
+----------+----------+--------------------------------------+----------+----------------------+
| Brain_11 | ACTIVE   | e13426e2-1f0d-4ee0-952a-ef6a7bcf40a2 | m1.small | 2014-09-17T18:02:21Z |
+----------+----------+--------------------------------------+----------+----------------------+
| Brain_9  | ACTIVE   | 81844eb9-1693-41e8-a976-e8e81ba477a1 | m1.small | 2014-09-16T18:20:03Z |
+----------+----------+--------------------------------------+----------+----------------------+
```

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

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).

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.
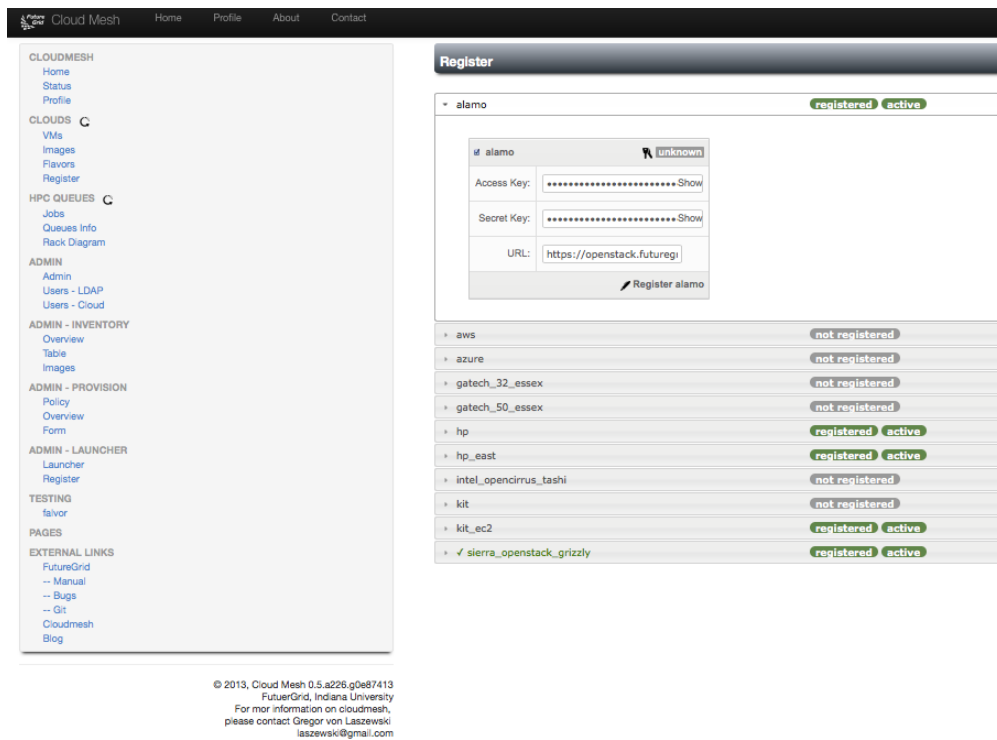
Figure 7.3: **Figure:** Registering an EC2 compatible cloud
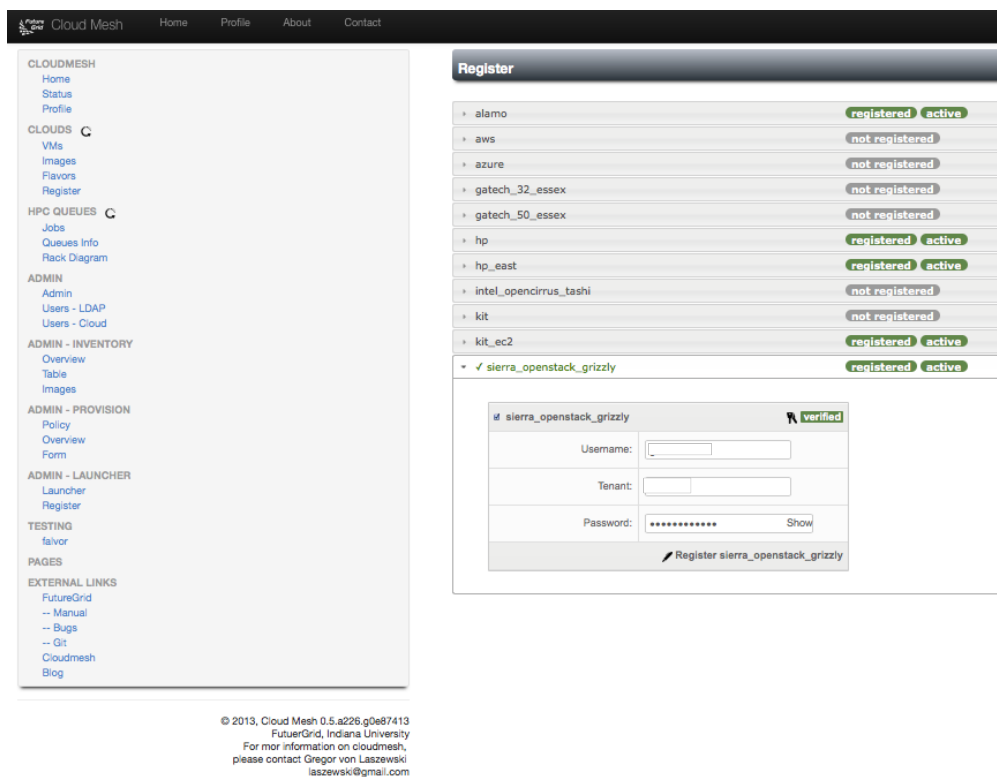
Figure 7.4: **Figure:** Registering an OpenStack protocol compatible cloud.

Figure 7.5: **Figure:** Starting and deletion of VMs is easy in cloudmesh through a simple table view.

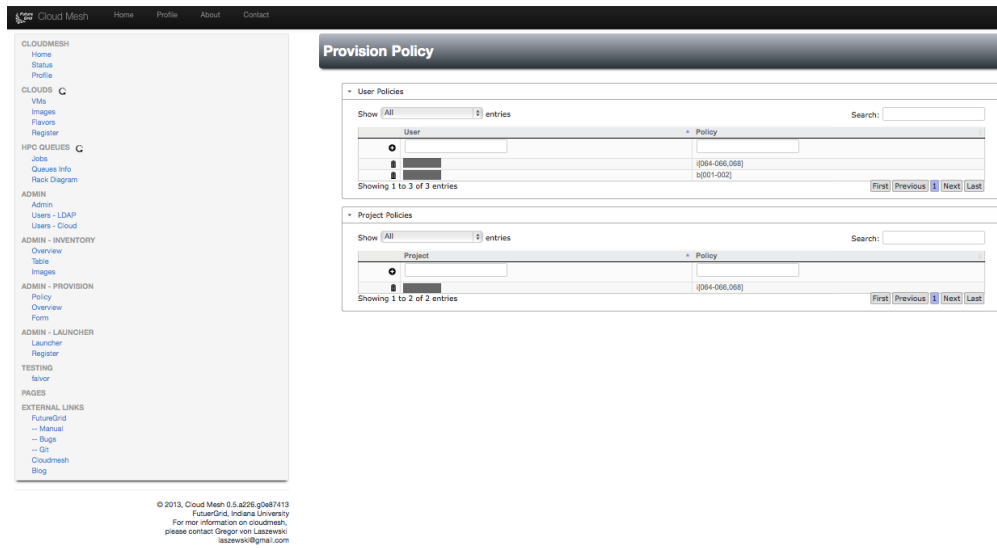Figure 7.6: **Figure:** Launching predefined configurations on FutureGrid

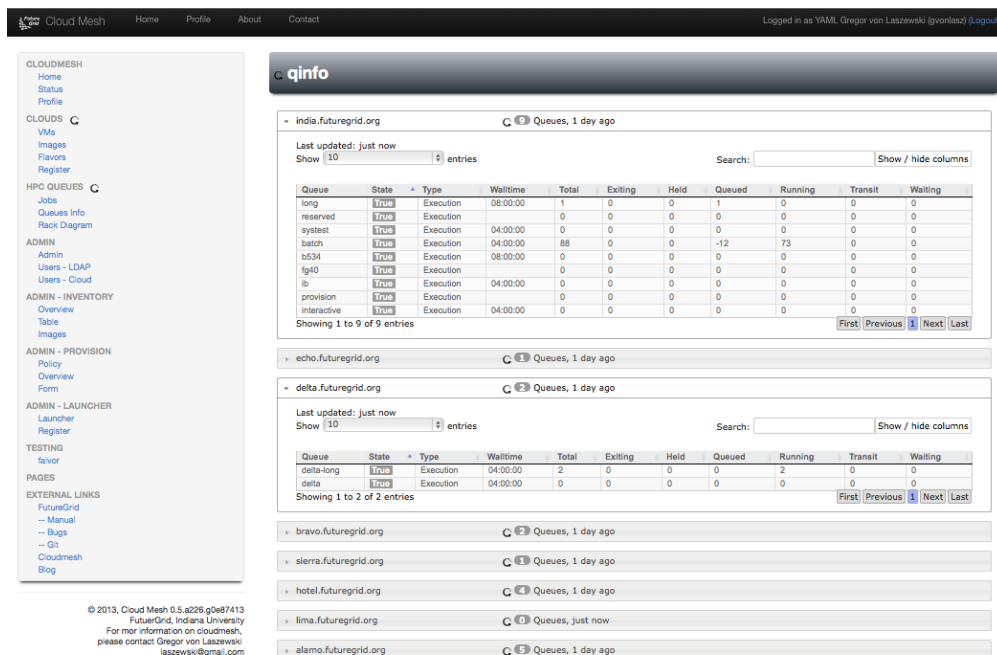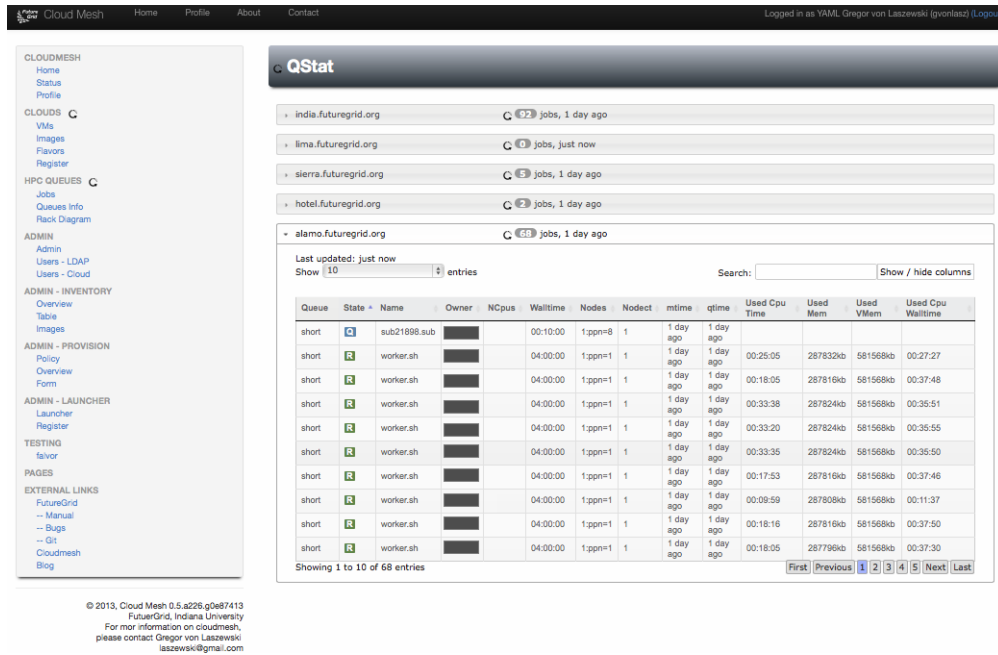Figure 7.7: **Figure:** Defining the baremetal access policy

Figure 7.8: **Figure:** Listing the available queues

Figure 7.9: **Figure:** Listing the queue information about jobs and status

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.
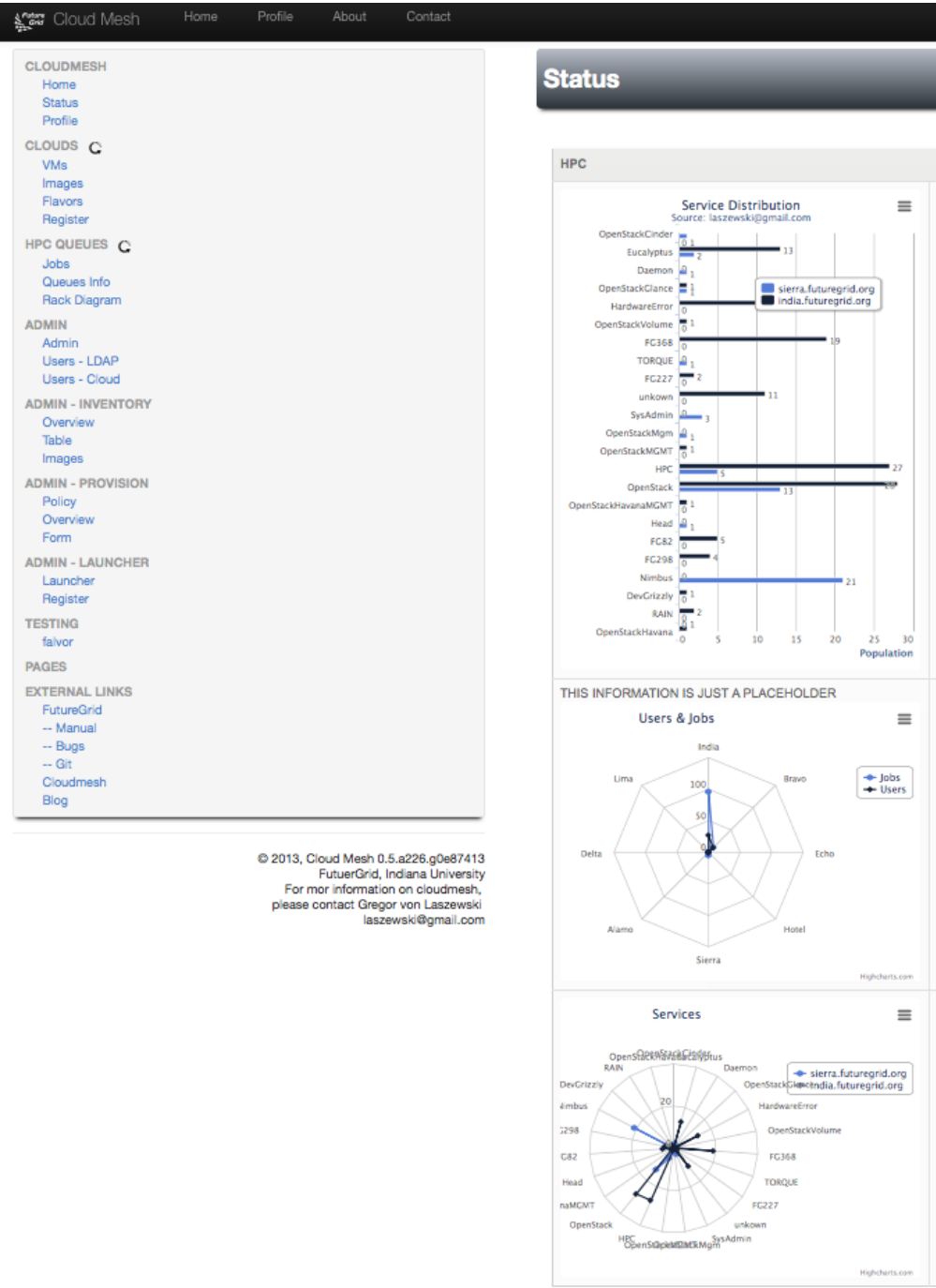
Figure 7.10: **Figure:** Displaying a simple status of the systems (here HPC).
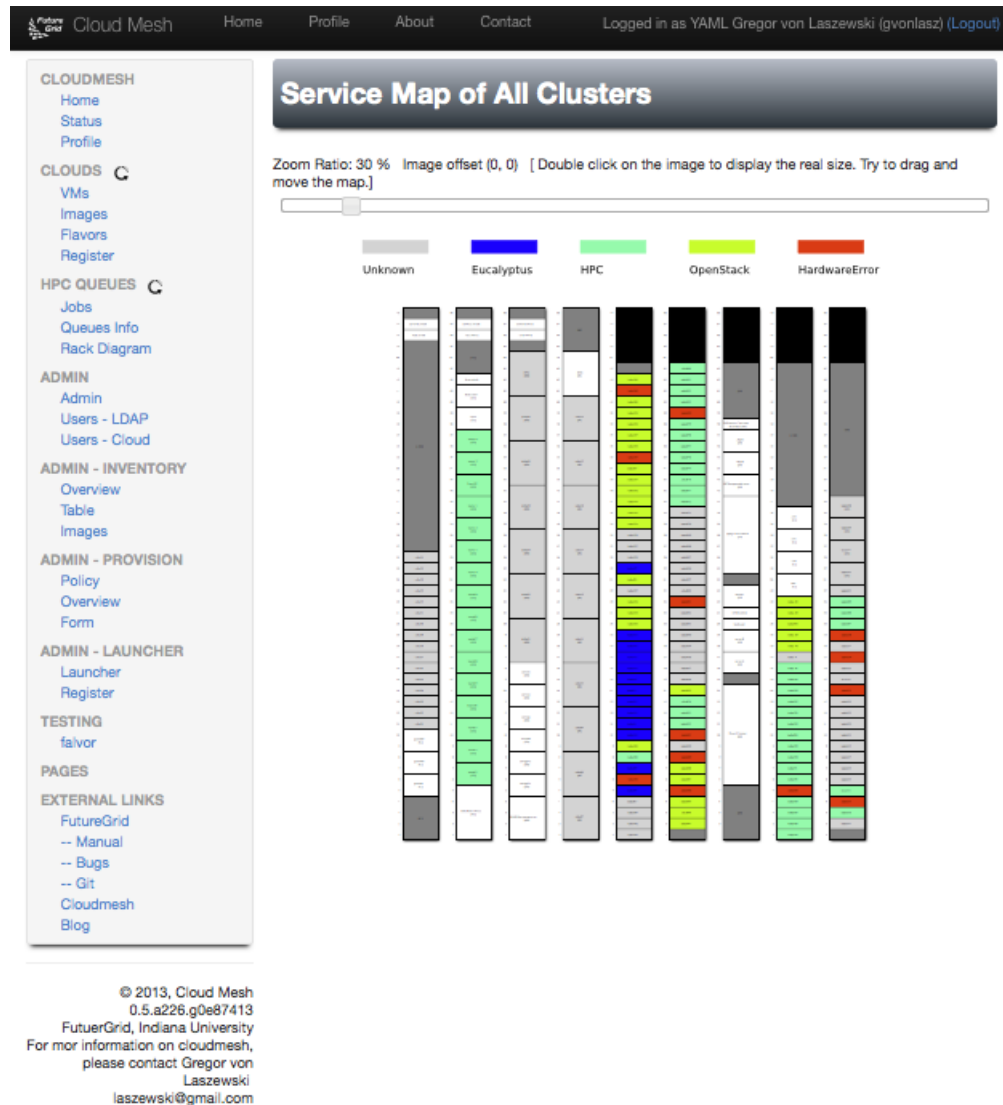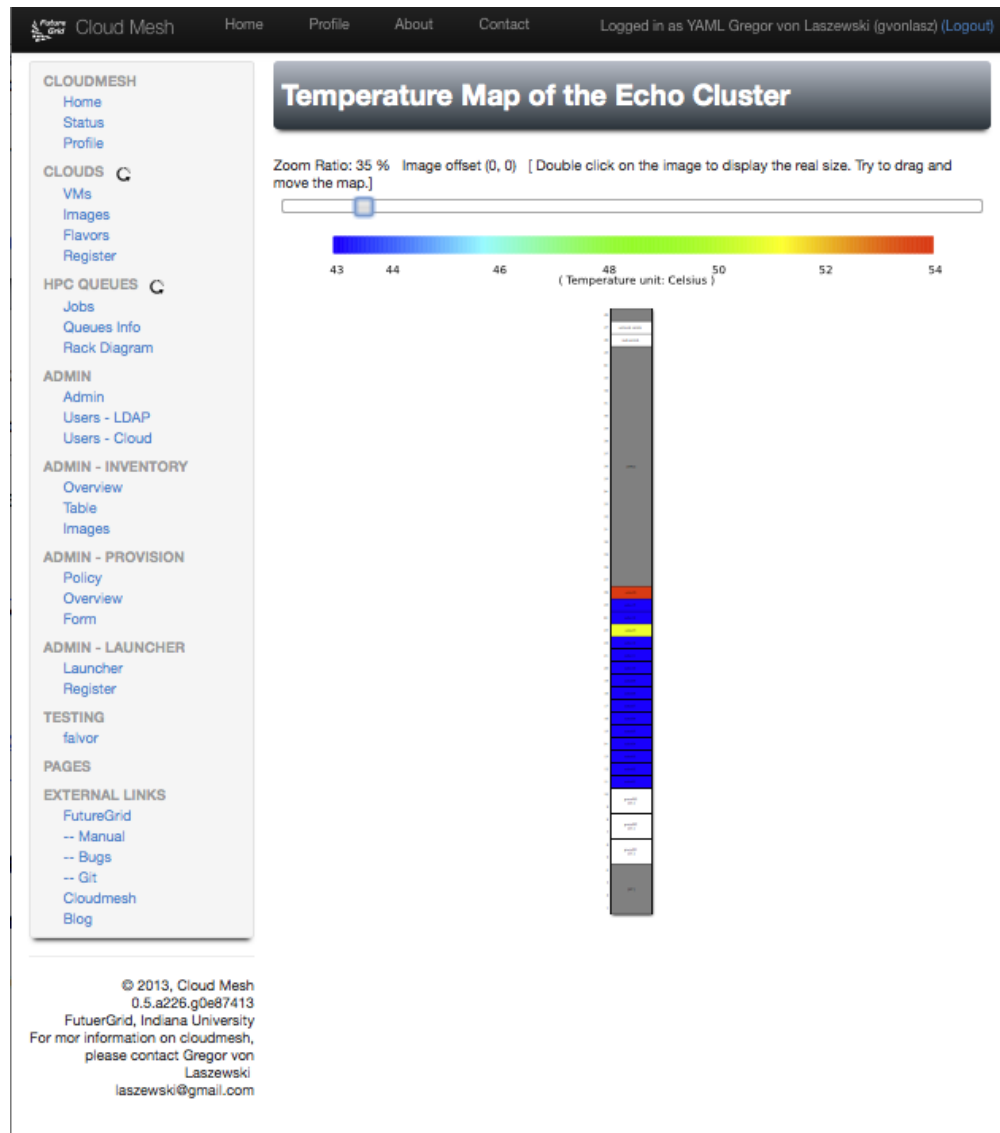
Figure 7.11: **Figure:** Inventory of the systems.

Figure 7.12: **Figure:** Service map to depict which server is dedicated to which services

Figure 7.13: **Figure:** Temperature map of a rack

EIGHT

PYTHON

8.1 Ecosystem

8.1.1 virtualenv

Often you have your own computer and you do not like to change its environment to keep it in prestine condition. Python comes with mnay libraries that could for example conflict with libraries that you have installed. To avoid this it is bets to work in an isolated python environment while using virtualenv,. Documentation about it can be found at:

```
* http://virtualenv.readthedocs.org/
```

The installation is simple once you have pip installed. If it is not installed you can say:

```
$ easy_install pip
```

After that you can install the virtual env with:

```
$ pip install virtualenv
```

To setup an isolated environment for example in the directory ~/ENV please use:

```
$ virtualenv ~/ENV
```

To activate it you can use the command:

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

you can put this command n your bashrc or bash_profile command so you do not forget to activate it.

8.1.2 pypi

The Python Package Index is a large repository of software for the Python programming language containing a large number of packages [link]. The nice think about pipy is that many packages can be installed with the program 'pip'.

To do so you have to locate the <package_name> for example with the search function in pypi and say on the com-mandline:

```
pip install <package_name>
```

where pagage_name is the string name of the package. an example would be the package called fabric which you can install with:

```
pip install fabric
```

If all goes well the package will be installed.

8.1.3 github

Github is a code repository that allows the development of code in a distributed fashion. There are many good tutorials about github.

Some of them can be found on the github Web page. An interactive tutorial is for example available at

- https://try.github.io/

A more extensive list of tutorials can be found at

-

https://help.github.com/articles/what-are-other-good-resources-for-learning-git-and-github

Important is that you always want to make sure that you want to use the git init command and add your Name and e-mail. Do it consistent in the machines you use, or your checkins in git (if you do them) do not show up in a consistant fashion as a single user. This is done with the following commands:

```
$ git config --global user.name "John Doe"
$ git config --global user.email johndoe@example.com
```

You can set also the editor with:

```
$ git config --global core.editor emacs
```

More information about a first time setup is documented at:

```
$ http://git-scm.com/book/en/Getting-Started-First-Time-Git-Setup
```

To check your setup you can say:

```
$ git config --list
```

8.2 Python resources

There is a rich set of resources related to python programming on the internet. Some resources that we found useful include:

- http://ivory.idyll.org/articles/advanced-swc/
- http://python.net/~goodger/projects/pycon/2007/idiomatic/handout.html
- http://www.youtube.com/watch?v=0vJJlVBVTFg

The following have been found useful by previous students. However there may be much better resources out there.

- http://www.korokithakis.net/tutorials/python/

and here is one that you can do on a web browser in case you have not installed python on your computer

- http://www.afterhoursprogramming.com/tutorial/Python/Introduction/

8.3 Lanaguage Features

8.3.1 Print

```
>>> print "Hello Cloud"
Hello Cloud
```

8.3.2 Comments

Simple comment

```
>>> # this is a comment
```

Multiline Comment with "

```
"""
this is a multiline comment
"""
```

Multiline Comment with '

```
'''
this is a multiline comment
'''
```

8.3.3 Variables

```
>>> flavor = "tiny"
>>> print flavor
tiny
```

Changing the variable Value

```
>>> flavor = "large"
>>> print flavor
large
```

8.3.4 List

```
>>> list = [1, 2, 3, 4, 5, 6]
>>> print list[0]
1
>>> print list[len(list)-1]
6
>>> print list[2:5]
[3, 4, 5]
```

8.3.5 For Loop

```
>>> list = [1, 2, 3, 4, 5, 6]
>>> for element in list:
...     print element
...
1
2
3
4
5
6

>>> flavors = ['tiny', 'medium', 'large']
>>> for flavor in flavors:
...     print flavor
...
tiny
medium
large
```

8.3.6 If Condition

```
>>> flavor = "tiny"
>>> if flavor == "tiny":
...     print("vanilla has a tiny flavor")
... elif flavor == "large":
...     print("large flavor")
... else:
...     print("this flavor I do not like")
...
vanilla has a tiny flavor
```

8.3.7 Arithmetic Operators

```
>>> list = [1, 2, 3, 4, 5, 6]
>>> print sum(list)
21
>>> print min(list)
1
>>> print max(list)
6
>>> print sum(list)/len(list)
3
>>> print sum(list)/float(len(list))
3.5
```

8.3.8 Functions

Function to do calculations

```
>>> def f(x,y):
...     return x+y+ y*y + x*x
...
>>> print f(1,2)
8
>>> print f(4,6)
62
```

Functions for String Manipulation

```
>>> def name(firstname, lastname):
...     return "%s %s" % (firstname, lastname)
...
>>> def reversename(firstname, lastname):
...     return "%s, %s" % (lastname, firstname)
...
>>> print name('Gregor', 'von Laszewski')
Gregor von Laszewski
>>> print reversename('Gregor', 'von Laszewski')
von Laszewski, Gregor
```

8.4 Excersises

her is the task i like you to do to reinforce the learning experience once you are done with the tutorial.

1. Write a program that uses loops over both x and y coordinates while x is in 1,2,3,4,5 and y is in 5,4,3,2,1 and prints the x and y coordinate

2. Write a program that sums up all values in x and y

3. Write a program just like the first task but does not print values where x is equal to 2 and y is equal to 4

4. Write a function that takes in a word and returns it in reverse order

5. Provide a program that uses dicts

6. Read up on classes and write a program that uses classes.

7. We will create an icecream machine that produces icecream in with tiny flavor, medium flavor and large flavor.

8. In addition the icecream cone will be wrapped into some paper that has an image on it. Images will be Penguin, Apple, Emperor, King

8.5 Code Listings

8.5.1 Booting a virtual machine

Boot with a shell command

```
#! /bin/sh -x

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

```
nova boot --flavor m1.small \
           --image "futuregrid/ubuntu-14.04" \
           --key_name $USERNAME-india-key $USERNAME-001
```

Boot with python calling a shell command

```python
#!/uer/bin/env python

import os
#from sh import nova

homedir = os.path.expanduser("~/")
username = homedir.split("/")[-2]

# to avoid installing sh
#print nova ("flavor-list")

#print nova ("keypair-add", "--pub-key", "~/.ssh/id_rsa.pub" "%s-india-key" % username)
#print nova ("boot", "--flavor", "m1.small", "--image", "futuregrid/ubuntu-12.04", "--key_name", "%s-

os.system("nova keypair-add --pub-key ~/.ssh/id_rsa.pub {0}-india-key".format(username) )
os.system('nova boot --flavor m1.small --image "futuregrid/ubuntu-14.04" --key_name {0}-india-key {0}
```

Boot with cloudmesh

There are two ways to boot a VM with cloudmesh other than cloudmesh GUI, you may choose cloudmesh API or CLI. For example, boot a VM with API:

```python
import cloudmesh

mesh = cloudmesh.mesh("mongo")

username = cloudmesh.load().username()

mesh.activate(username)

mesh.start(cloud="india", cm_user_id=username flavor="m1.small", image="futuregrid/ubuntu-14.04")
```

boot a VM with CLI: (note here we use a convenient way to call cloudmesh shell commands by using cloudmesh.shell, so that we can illustrate our CLI in ipython notebook)

```python
cloudmesh.shell("vm start --cloud=india --flavor=m1.medium --image=futuregrid/ubuntu-12.04")
```

There are other ways to boot VMs with CLI, for more details about CLI command vm:

```python
print cloudmesh.shell("vm -h")
```

Deleting virtual machines

Delete with a shell command

```bash
#! /bin/bash

if [ "$1" != "" ]; then
```

```
    nova delete $1
else
    echo "VM ID or Name is required. Run 'nova list' to get the list of VMs"
    exit 1
fi
```

Delete with python calling a shell command

```
'''
# from sh import nova

from cloudmesh import vm_name

username = "gregor"

# simulator = False
simulator = True


if simulator:
    def nova (*args, **kwargs):
        return args
else:
    from sh import nova


for index in range(1, n + 1):
    print "DELETE VM", index, vm_name.(username, index)
    print nova ("delete", vm_name(username, index))
'''

#!/usr/bin/env python

import sys
import os

if len(sys.argv) != 2:
    print "VM ID or Name is required. 'nova list' will show the list of currently running VMs"
    sys.exit(1)
else:
    os.system("nova delete {0}".format(sys.argv[1]))
```

Delete with cloudmesh

Similar as booting a VM, we can also delete a VM with API and CLI of cloudmesh. For example, delete a VM with API: (note here you need to provide the ID of VM to delete. If you want to test this, please follow the initialization steps of the previous section 'boot with cloudmesh' first)

```
mesh.delete(cloud="india", server="provvide vm id", username)
```

delete a VM with CLI:

```
cloudmesh.shell("vm delete --id=provvide vm id --cloud=india --force")
```

There are other ways to delete VMs with CLI, for more details about CLI command vm:

```
print cloudmesh.shell("vm -h")
```

8.5.2 Query to Cloud

List Flavors

API:

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

CLI:

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

List Images

API:

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

CLI:

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

List Virtual Machines

API:

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

CLI:

```
cloudmesh.shell("list vm india --refresh")
```

8.6 TO INTEGARTE

8.6.1 cloud0-satrt-vm-simulator.py

```python
import sys
# from sh import nova
from cloudmesh import vm_name

username = "gregor"

# simulator = False
simulator = True

if simulator:
    def nova (*args, **kwargs):
        return args
else:
    from sh import nova
```

```python
n = 3

for index in range(1, n + 1):
    print "START VM", index
    try:
        print nova ("boot",
                    "--flavor", "m1.small",
                    "--image", "futuregrid/ubuntu-12.04",
                    "--key_name", "~/.ssh/id_rsa.pub",
                    vm_name(username, index))
    except:
        print "machine could not be started because of an error", name.format(username, index)

for index in range(1, n + 1):
    print "DELETE VM", index
    print nova ("delete", vm_name(username, index))
```

8.6.2 cloud1.py

```python
from sh import nova
from sh import grep
from pprint import pprint

print 70 * "-"
a = nova("list")
print a
print 70 * "-"
b = grep(a, "|")
print b
print 70 * "-"
c = b.split("\n")
print c
print 70 * "-"

lines = grep (nova("list"), "|").split("\n")

pprint (lines)
print " I am there"

lines = lines[1:-1]

pprint (lines)

print " I am here"
vms = {}


print 70 * "="
pprint(lines)
print 70 * "="

for line in lines:
    print "LINE", line
    line = line[2:-2]
    (id, name, status, networks) = line.split("|")
```

```
    print "-%s-" % id
    print "-%s-" % name
    vms[id] = {"id": id.strip(), "name":name.strip(), "status":status.strip(), "networks": networks.s

pprint(vms)
```

8.6.3 cloud2.py

```
from sh import nova
from sh import grep
from pprint import pprint
import webbrowser
import random
import math
import os

print 70 * ``-''
a = nova(``list'')
print a
print 70 * ``-''
b = grep(a,''|'')
print b
print 70 * ``-''
c = b.split(``\n'')
print c
print 70 * ``-''

lines = grep (nova(``list''), ``|'').split(``\n'')

pprint (lines)
print '' I am there''

lines = lines[1:-1]

pprint (lines)

print '' I am here''
vms = {}



print 70 * ``=''
pprint(lines)
print 70 * ``=''
alist = []
for line in lines:
    print ``LINE'', line
    line = line[2:-2]
    (id, name, status, networks) = line.split(``|'')
    print ``-%s-'' % id
    print ``-%s-'' % name
    vms[id] = {``id'': id.strip(), ``name'':name.strip(), ``status'':status.strip(), ``netw
    alist.append(status.strip())
```

```
pprint(vms)


class table:

    #self.filname
    #self.alist
    #self.cols
    #self.table
    #self.n
    #self.css

    def __init__(self, filename, cols,n,css):
        self.filename = filename
        self.cols = cols
        self.alist = []
        self.table = ``''
        self.n = n
        self.css= css

    def assign(self,alist):
        self.alist = alist

    def generate (self):
        self.table = ``<table border=1>\n''
        rows = int(math.ceil(len(self.alist)/float(self.cols)))
        item = 0
        for row in range(0,10):
            self.table = self.table + ``''
            for col in range(0,10):
                value = ``''
                if item < len(self.alist):
                    value =self.alist[item]
                    self.table = self.table + `<td class=''{0}''>{1}</td>'.format(value,val
                    item = item + 1
            self.table = self.table + ``</tr>\n''
        self.table = self.table + ``</table>''


    def write(self):
        f = open (self.filename, ``w'')
        print >>f, self.css
        print >>f, self.table
        f.close()



    def display(self):
        handle = webbrowser.get()
        handle.open(self.filename)

    def random(self):
        states =['active', `build', `error']
        self.alist = []
```

```
        for i in range(0,self.n):
            value = random.randint(0,2)
            print value
            self.alist.append(states[value])

filename = `t.html'
css = ``'''``<style>
    .build {
        background-color:red;

    }
    .active {
        background-color:blue;
    }

    .error {
        background-color:green;

    }
</style> ``''''

#os.remove(filename)


nelem = len(alist)
t = table(filename, 10,nelem,css)
t.assign(alist)

#t.random()
t.generate()
t.write()
t.display()



#w = table(filename, 7,20)

#w.random()
#w.generate()
#w.write()
#w.display()
```

8.6.4 cloud3.py

```
import sys
import json
from sh import nova
from sh import grep
from pprint import pprint
```

```python
class cloud:

    def __init__(self):
        self.vms = {}
        self.status = []

    def refresh(self):
        lines = grep (nova("list"), "|").split("\n")[1:-1]
        self.status = []

        for line in lines:
            line = line[2:-2]
            (id, name, status, networks) = line.split("|")
            self.vms[id] = {"id": id.strip(),
                       "name":name.strip(),
                       "status":status.strip(),
                       "networks": networks.strip()}

    def get_status(self):
        self.status = []
        for vm in self.vms:
            self.status.append(self.vms[vm]["status"])
        return self.status

    def __str__(self):
        return json.dumps(self.vms,indent=4)

def main():
    c = cloud ()
    c.refresh()

    pprint(c.get_status())
    pprint(c)

if __name__ =='__main__':
    main()
```

8.6.5 cloud4.py

```python
import time
import webbrowser
from pprint import pprint
from sh import grep, nova
import json
import sys

simulator = False
#simulator = True


if simulator:
    def nova (*args, **kwargs):
        return args
else:
    from sh import nova
```

```python
class cloud:

    def __init__(self):
        self.vms = {}

    def set (self,vms):
        self.vms = vms

    def refresh(self):
        self.vms = {}
        lines = grep (nova("list"), "|").split("\n")[1:-1]
        for line in lines:
            line = line[2:-2]
            (id, name, status, networks) = line.split("|")
            self.vms[id] = {"id": id.strip(),
                    "name":name.strip(),
                    "status":status.strip(),
                    "networks": networks.strip()}

    def table(self, columns, name="__undefined__"):
        #columns = ["status", "name"]
        table_s = "<hr>\n"
        table_s += "<h1> My Cloud </1>"
        table_s += "<hr>\n"
        table_s += '<table border="1" cellpadding="10">\n'
        table_s += "\t<tr>\n\t\t<th>"
        table_s += "</th>\n\t\t<th>".join(columns)
        table_s += "</th>\n\t</tr>\n"
        for vm in self.vms:
            table_s += "\t<tr>\n"
            attribute_list = []
            for attribute in columns:
                value = self.vms[vm][attribute]

                if attribute == "status":
                    if value == "ACTIVE":
                        color = 'style="background-color:green"'
                    elif value == "ERROR":
                        color = 'style="background-color:red"'
                    else:
                        color = 'style="background-color:blue"'

                elif attribute == "name":
                    if value.startswith(name):
                        color = 'style="background-color:green"'
                    else:
                        color = ''

                else:
                    color = ""

                table_s += "\t\t<td {0}>{1}</td>\n".format(color,value)
            table_s += "\n\t</tr>\n"
        table_s += "</table>\n"
        return table_s

    def status(self):
```

```python
        status_list = []
        for vm in self.vms:
            status_list.append(self.vms[vm]["status"])
        return status_list

    def __str__(self):
        return json.dumps(self.vms, indent=4)

    def display(self, columns=None,name="__undefined__"):
        filename = "table.html"
        f = open (filename, "w")
        print >> f, self.table(columns,name)
        f.close()
        handle = webbrowser.get()
        handle.open(filename)

    def start(self, name):
        result = nova ("boot",
                       "--flavor", "m1.small",
                       "--image", "futuregrid/ubuntu-12.04",
                       "--key_name", "~/.ssh/id_rsa.pub",
                       name)
        return result

    def delete(self,name):
        results = nova ("delete", name)
        return results

    def  start_n (self, name, n):
        """fill this out"""
        length = len (str(n))
        name_format = "{0}-{1:0" + str(length) + "d}"

        print "start", n, "vm(s) with name", name
        for index in range(1, n + 1):
            print "START VM", index
            try:
                print nova ("boot",
                            "--flavor", "m1.small",
                            "--image", "futuregrid/ubuntu-12.04",
                            "--key_name", "~/.ssh/id_rsa.pub",
                            name_format.format(name, index))
            except:
                print "machine could not be started because of an error", name.format(username, index)

    def delete_n(self,name,n):
        length = len (str(n))
        name_format = "{0}-{1:0" + str(length) + "d}"

        for index in range(1, n + 1):
            print "DELETE VM", index
            print nova ("delete",
                        name_format.format(name, index))

    def commandline(self):

        # write out the commands
        # commnd:
```

```python
        # s 1  starts vm with index 1
        # s 3  starts vm with index 3
        # d 1 dletes vm with indes 1
        # d 3 dletes vm with indes 1
        # v updates the browser window and displays the curren t status

        # bonus

        # s 2 10  starts 10 vms starting from index 2
        # d 2 10  starts 10 vms starting from index 2

        while True:
            line = raw_input("cloud> ")
            if line =="q":
                break
            else:
                print "you typed", line
                ##### do all the programming here
                if line.startswith("v"):
                    self.refresh()
                    self.display(["status", "name", "id"], username)
                else:
                    command = line.split(" ")
                    print command
                    if command[0] == "d":
                        if len(command) == 2:
                            print "start", command
                            name = command[1]
                            result = self.delete(name)
                            print result

                        elif len(command) == 3:
                            print "delete", command
                            name = command[1]
                            n = int(command[2])
                            self.delete_n(name,n)

                        else:
                            print " wrong number of arguments"
                    elif command[0] == "s":
                        if len(command) == 2:
                            print "excute", command
                            name = command[1]
                            result = self.start(name)
                            print result
                        elif len(command) == 3:
                            print "bleep", command
                            name = command[1]
                            n = int(command[2])
                            self.start_n(name,n)
                        else:
                            print " wrong number of arguments"


        pass

if __name__ == '__main__':
```

```python
    cloud_dict = {
        "317aa10b-96e8-4705-a725-2b6deccd8535 ": {
            "status": "ACTIVE",
            "networks": "private=10.35.23.96",
            "id": "317aa10b-96e8-4705-a725-2b6deccd8535",
            "name": "a2"
            },
        "59e5ee03-4e94-4265-8008-ba1d3b460ba7 ": {
            "status": "ERROR",
            "networks": "private=10.35.23.52",
            "id": "59e5ee03-4e94-4265-8008-ba1d3b460ba7",
            "name": username + "001"
            },
        "e0926229-b252-4cac-93a8-2e614ef0a2cf ": {
            "status": "BUILDING",
            "networks": "private=10.35.23.94",
            "id": "e0926229-b252-4cac-93a8-2e614ef0a2cf",
            "name": "a1"
            }
        }


    c = cloud ()
    c.commandline()
    sys.exit()
    name = "booh"

    c.start_n(name,2)
    c.refresh()
    c.display(["status", "name", "id"], username)
    time.sleep(3)
    c.refresh()
    c.display(["status", "name", "id"], username)


    c.delete_n(name,2)
    time.sleep(10)
    c.refresh()
    c.display(["status", "name", "id"], username)


#


#   c.refresh()
# c.display(["status", "name", "id"], username)
#   time.sleep(1)

#   c.refresh()
#   c.display(["status", "name", "id"], username)
#   time.sleep(1)

#   c.refresh()
#   c.display(["status", "name", "id"], username)
#   time.sleep(1)

#   c.delete(name)
#   time.sleep(5)
```

```
#     c.refresh()
#     c.display(["status", "name", "id"], username)

# c.display()


# print c.status()
# c.set(cloud_dict)
# print c
# print c.table(["status", "name"], username)
```

8.6.6 tabel2.py

```
import webbrowser
import random
import math

class table:


    def make_table(cols, list):

        table = ``<table border=1>\n''

        rows = int(math.ceil(len(list) / float(cols)))

        item = 0
        for row in range(0, 10):

                table = table + ''  <tr>''

                for col in range(0, 10):
                        if item < len(list):
                                        value = list[item]
                        table = table + ``<td>{0}</td>''.format(value)
                        item = item + 1
                        table = table + ``</tr>\n''

                table = table + ``</table>'' + ``%i'' % item



def write_table(filename, table):
        f = open (`abc.html', `w')
        print >> f, table

def display_table(filename):
        handle = webbrowser.get()
        handle.open(`abc.html')

n = 91
list = []
```

```
for i in range(0, n):
        list.append(random.randint(1, 10))
print len(list), list
filename = ``abc.html''
table = make_table(10, list)
write_table(filename, table)
display_table(filename)

print table
```

8.6.7 table1.py

```python
table = "<table border=1>\n"

item = 1

for row in range(0, 10):
    table = table + "  <tr>"

    for col in range(0, 10):
        table = table + "<td>"
        if item <= 97:
            table = table + "%i" % item
        item = item + 1
        table = table + "</td>"

    table = table + "</tr>\n"

table = table + "</table>"

f = open ('abc.html', 'w')
print >> f, table

import webbrowser
handle = webbrowser.get()
handle.open('abc.html')
```

8.6.8 table3.py

```python
import webbrowser
import random
import math

class table:


    def make_table(cols, list):
        table = ``<table border=1>\n''
        rows = int(math.ceil(len(list) / float(cols)))
        item = 0
        for row in range(0, 10):
            table = table + ''  <tr>''
        for col in range(0, 10):
```

```
                    value = ``''
          if item < len(list):
              value = list[item]
              table = table + ``<td>{0}</td>''.format(value)
              item = item + 1
              table = table + ``</tr>\n''
              table = table + ``</table>'' + ``%i'' % item

    def write_table(filename, table):
        f = open (`abc.html', `w')
        print >> f, table

    def display_table(filename):
        handle = webbrowser.get()
        handle.open(`abc.html')

print table

n = 91
list = []

for i in range(0, n):
    list.append(random.randint(1, 10))
print len(list), list
filename = ``abc.html''
table = make_table(10, list)
write_table(filename, table)
display_table(filename)
```

8.6.9 table4.py

```python
import webbrowser
import random
import math


class table:

    # self.filname

    def __init__(self, filename):
        self.filename = filename

    def display (self, cols, list):
        table = "<table border=1>\n"
        rows = int(math.ceil(len(list) / float(cols)))
        item = 0
        for row in range(0, 10):
            table = table + "  <tr>"
        for col in range(0, 10):
            value = ""
        if item < len(list):
            value = list[item]
            table = table + "<td>{0}</td>".format(value)
```

```
            item = item + 1
            table = table + "</tr>\n"
            table = table + "</table>" + "%i" % item

    def read(self, filename, table):
        f = open ('abc.html', 'w')
        print >> f, table

    def write(self, filename):
        handle = webbrowser.get()
        handle.open('abc.html')

print

n = 91
list = record()
r.input()
r.output()

for i in range(0, n):
    list.append(random.randint(1, 10))
print len(list), list
filename = "abc.html"
r = record(10, list)
r.input(filename, table)
r.output(filename)
```

8.6.10 table5.py

```
import webbrowser
import random
import math


class table:

    # self.filname


    # self.table

    def __init__(self, filename, cols, alist):
        self.filename = filename
        self.cols = cols
        self.alist = alist
        self.table = ""

    def generate (self):
        self.table = "<table border=1>\n"
        rows = int(math.ceil(len(self.alist) / float(self.cols)))
        item = 0
        for row in range(0, 10):
            self.table = self.table + "  <tr>"
            for col in range(0, 10):
                value = ""
                if item < len(alist):
```

```python
            value = alist[item]
            self.table = self.table + "<td>{0}</td>".format(value)
            item = item + 1
        self.table = self.table + "</tr>\n"
    self.table = self.table + "</table>" + "%i" % item

    def write(self):
        f = open (self.filename, 'w')
        print >> f, self.table
        f.close()

    def display(self):
        handle = webbrowser.get()
        handle.open(self.filename)

    def random(self, n):
        self.random = ""
        print random

print

n = 91
alist = []
for i in range(0, n):
    alist.append(random.randint(1, 10))

t = table('abc.html', 10, alist)

t.random(91)
t.generate()
t.write()
t.display()
```

8.6.11 table6.py

```python
import webbrowser
import random
import math
import os

class table:

    # self.filname
    # self.alist
    # self.cols
    # self.table
    # self.n
    def __init__(self, filename, cols, n):
        self.filename = filename
        self.cols = cols
        self.alist = []
        self.table = ""
        self.n = n

    def generate (self):
        self.table = "<table border=1>\n"
```

```
        rows = int(math.ceil(len(self.alist) / float(self.cols)))
        item = 0
        for row in range(0, 10):
            self.table = self.table + "  <tr>"
            for col in range(0, 10):
                value = ""
                if item < len(self.alist):
                    value = self.alist[item]
                    self.table = self.table + "<td>{0}</td>".format(value)
                    item = item + 1
            self.table = self.table + "</tr>\n"
        self.table = self.table + "</table>" + "%i" % item

    def write(self):
        f = open (self.filename, "a")
        print >>f, self.table
        f.close()


    def display(self):
        handle = webbrowser.get()
        handle.open(self.filename)

    def random(self):

        self.alist = []
        for i in range(0,self.n):
            self.alist.append(random.randint(1,10))

filename = 't.html'
os.remove(filename)

t = table(filename, 10,91)

t.random()
t.generate()
t.write()
t.display()

w = table(filename, 7,20)

w.random()
w.generate()
w.write()
w.display()
```

8.6.12 table7.py

```
import webbrowser
import random
import math
import os

class table:

    # self.filname
```

```
    #self.alist
    #self.cols
    #self.table
    #self.n
    #self.css

    def __init__(self, filename, cols,n,css):
        self.filename = filename
        self.cols = cols
        self.alist = []
        self.table = ``''
        self.n = n
        self.css= css

    def generate (self):
        self.table = ``<table border=1>\n''
        rows = int(math.ceil(len(self.alist)/float(self.cols)))
        item = 0
        for row in range(0,10):
            self.table = self.table + ``''
            for col in range(0,10):
                value = ``''
                if item < len(self.alist):
                    value =self.alist[item]
                    self.table = self.table + `<td class=''{0}''>{1}</td>'.format(value,va
                    item = item + 1
            self.table = self.table + ``</tr>\n''
        self.table = self.table + ``</table>''


    def write(self):
        f = open (self.filename, ``w'')
        print >>f, self.css
        print >>f, self.table
        f.close()



    def display(self):
        handle = webbrowser.get()
        handle.open(self.filename)

    def random(self):
        states =['active', `build', `error']
        self.alist = []
        for i in range(0,self.n):
            value = random.randint(0,2)
            print value
            self.alist.append(states[value])

filename = `t.html'
css = ``''``<style>
    .build {
        background-color:red;
```

```
    }
    .active {
        background-color:blue;
    }

    .error {
        background-color:green;

    }
</style> ``''''

#os.remove(filename)

t = table(filename, 10,91,css)

t.random()
t.generate()
t.write()
t.display()



#w = table(filename, 7,20)

#w.random()
#w.generate()
#w.write()
#w.display()
```

8.6.13 table8.py

```python
import webbrowser
import random
import math
import os

class table:

    # self.filname
    # self.alist
    # self.cols
    # self.table
    # self.n
    def __init__(self, filename, cols, n):
        self.filename = filename
        self.cols = cols
        self.alist = []
        self.table = ""
        self.n = n
```

```python
    def generate (self):
        self.table = "<table border=1>\n"
        rows = int(math.ceil(len(self.alist) / float(self.cols)))
        item = 0
        for row in range(0, 10):
            self.table = self.table + "  <tr>"
            for col in range(0, 10):
                value = ""
                if item < len(self.alist):
                    value = self.alist[item]
                    self.table = self.table + "<td>{0}</td>".format(value)
                    item = item + 1
            self.table = self.table + "</tr>\n"
        self.table = self.table + "</table>" + "%i" % item

    def write(self):
        f = open (self.filename, "a")
        print >> f, self.table
        f.close()



    def display(self):
        handle = webbrowser.get()
        handle.open(self.filename)

    def random(self):

        self.alist = []
        for i in range(0, self.n):
            self.alist.append(random.randint(1, 10))
filename = 't.html'
os.remove(filename)

t = table(filename, 10, 91)

t.random()
t.generate()
t.write()
t.display()

w = table(filename, 7, 20)

w.random()
w.generate()
w.write()
w.display()
```

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: FutureGrid will be soon changing its name to 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.

Login

Currently FutureSystems OpenStack Havana installed on India.

To use it you need to first log into india and prepare your Openstack credentials (Make sure to replace the 'username' with your actual FG username):

```
$ ssh username@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 |        |
+--------------------------------------+------------------------+--------+--------+
```

> **Hint**
>
> $USER is your username on the machine.

Key management

To start a virtual machine you must first upload a key to the cloud. This can be easily done in the following way:

```
$ nova keypair-add $USER-key > ~/.ssh/$USER-key
$ chmod 600 ~/.ssh/$USER-key
$ nova keypair-list
+--------------+-------------------------------------------------+
| Name         | Fingerprint                                     |
+--------------+-------------------------------------------------+
```

```
| <USER>-key    | ab:a6:63:82:dd:08:d3:bc:c0:21:56:4c:e2:bb:22:ac |
+--------------+-------------------------------------------------+
```

Where USER is your login name on india.

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
```

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 $USER-key $USER-001
```

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                   | <USER>-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              |                                      |
+---------------------------+--------------------------------------+
```

Where USER is your login name on india.

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           |
+--------------------------------------+------------+--------+--------------------+
| e15ad5b6-c3f0-4c07-996c-3bbe709a63b7 | <USER>-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/$USER-key 10.35.23.18
```

If you see a warning similar to:

```
Add correct host key in /home/<USER>/.ssh/known_hosts to get rid of this message.
Offending key in /home/<$USER>/.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 t is still there to be reused. To create one 5G volume you can do

```
$ nova volume-create 5
```

To list the volumes you can use:

```
$ nova volume-list
+--------------------------------------+-----------+--------------+------+-------------+-------------
| ID                                   | Status    | Display Name | Size | Volume Type | Attached to
+--------------------------------------+-----------+--------------+------+-------------+-------------
| 6d0d8285-xxxx-xxxx-xxxx-xxxxxxxxxxxx | available | None         | 5    | 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 $USER-001 6d0d8285-xxxx-xxxx-xxxx-xxxxxxxxxxxx "/dev/vdb"
```

Hint

Note thate $USER-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/$USER-key 10.35.23.18
ubuntu@<USER>-001:~$ sudo su -
root@<USER>-001:~# mkfs.ext4 /dev/vdb
root@<USER>-001:~# mount /dev/vdb /mnt
root@<USER>-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  4.2G   1% /mnt
```

When you want to detach it, unmount /mnt first, go back to indias's login node and execute volume-detach:

```
root@<USER>-001:~# umount /mnt
root@<USER>-001:~# exit
ubuntu@<USER>-001:~$ exit

$ nova volume-detach $USER-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 $USER-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 $USER 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 $USER-001 fg101/$USER/my-ubuntu-01
$ nova image-list
+-------------------------------------+----------------------------+--------+----------------------
| ID                                  | Name                       | Status | Server
+-------------------------------------+----------------------------+--------+----------------------
| 18c437e5-d65e-418f-a739-9604cef8ab33 | futuregrid/fedora-18      | ACTIVE |
| 1a5fd55e-79b9-4dd5-ae9b-ea10ef3156e9 | futuregrid/ubuntu-14.04   | ACTIVE |
| f43375b4-44d3-4350-a9a8-a73f35589344 | fg101/<USER>/my-ubuntu-01 | ACTIVE | c0bd849a-221a-4e53-bf7
+-------------------------------------+----------------------------+--------+----------------------
```

If you want to download your customized image, you can do it with this:

```
$ glance image-download --file "my-ubuntu-01.img" "fg101/$USER/custom-ubuntu-01"
```

> **Hint**
>
> 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 $USER-key \
         --user-data mycloudinit.txt $USER-002
```

You should be able to login to <USER>-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/$USER/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 $USER-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. One such tool are the 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
```
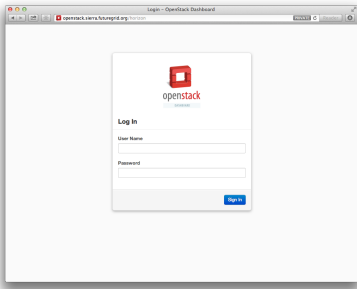

Note: Here's our known issues on using euca2ools or ec2 interface.

- 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 *s-openstack-grizzly*. **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.

PAAS

10.1 Using Hadoop in FutureGrid

Screencast

A screenncast of a subset of the information presented her is available at [YouTube] PC8h1CtVzH4.

We have various platforms that support Hadoop on FutureGrid. 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 FutureGrid using the MyHadoop tool.

myHadoop on FutureGrid

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 FutureGrid systems.

Login into a machine tha has myHadoop installed

To run the example we need to firts log into a FutureGrid 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 FutureGrid 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 FutureGrid password, but the One Time Password generated from your OTP token. Do not type your FutureGrid 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 FutureGrid portal.

Load the needed modules

Next, we need to load the myHadoop module. On some FutureGrid 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.

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 *s-accounts*. 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.:

```
1   #! /bin/bash
2
3   # OPTIONS FOR THE SCRIPT
4   #PBS -M username@example.com
5   #PBS -N ring_test
6   #PBS -o ring_$PBS_JOBID.out
7   #PBS -e ring_$PBS_JOBID.err
8   #PBS -q batch
9   #PBS -l nodes=4:ppn=8
10  #PBS -l walltime=00:20:00
11
12
13  # make sure MPI is in the environment
14  module load openmpi
15
16  # launch the parallel application with the correct number of process
17  # Typical usage: mpirun -np <number of processes> <executable> <arguments>
18  mpirun -np 32 ring -t 1000
19
20  echo "Nodes allocated to this job: "
21  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.

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 force10-s60
Bravo	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)	

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.

13.5 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

13.6 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

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

14.1 Sections

RST allows to specify a number of sections. You can do this with the various underlines:

```
*********************
Chapter
*********************
Section
=====================
Subsection
---------------------
Subsubsection
^^^^^^^^^^^^^^^^^^^^^^
Paragraph
~~~~~~~~~~~~~~~~~~~~~~
```

14.2 Listtable

```
.. csv-table:: Eye colors
   :header: "Name", "Firstname", "eyes"
   :widths: 20, 20, 10

   "von Laszewski", "Gregor", "gray"
```

Table 14.1: a title

Name	Firstname	eyes
von Laszewski	Gregor	gray

14.3 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

14.4 Boxes

14.4.1 Seealso

```
.. seealso:: This is a simple **seealso** note.
```

See also:

This is a simple **seealso** note.

14.4.2 Note

Note: This is a **note** box.

```
.. note::  This is a **note** box.
```

14.4.3 Warning

> **Warning:** note the space between the directive and the text

```
.. warning:: note the space between the directive and the text
```

14.4.4 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.
```

14.5 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.

14.6 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
Fri Sep 19 15:20:26 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.

14.7 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

14.8 Todo

```
.. todo:: an example
```

Todo

an example

FIFTEEN

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, please proovide an extension to this function to

(The *original entry* is located in /Users/flat/github/introduction_to_cloud_computing/docs/source/cloudmesh/cm/_cm-project.rst, line 206.)

Todo

Hyungro, please add the –format=FORMAT flag to print the

(The *original entry* is located in /Users/flat/github/introduction_to_cloud_computing/docs/source/cloudmesh/cm/_cm-project.rst, line 219.)

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/_cm_vm_management. line 133.)

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/_cm_vm_management. line 359.)

Todo

---

get the cloud credentials form a yaml file

(The *original entry* is located in /Users/flat/github/introduction_to_cloud_computing/docs/source/sh/_vm_sh.rst, line 7.)

Todo

use the information to start a VM

(The *original entry* is located in /Users/flat/github/introduction_to_cloud_computing/docs/source/sh/_vm_sh.rst, line 9.)

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/sh/_vm_sh.rst, line 11.)

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*
- *notebooks*