cloudmesh_client Documentation

Release 1.1.5

Gregor von Laszewski

February 11, 2016

The *cloudmesh client* toolkit is a lightweight client interface of accessing heterogeneous clouds, clusters, and workstations right from the users computer. The user can manage his own set of resources he would like to utilize. Thus the user has the freedom to customize their cyber infrastructure they use. Cloudmesh client includes an API, a commandline client, and a commandline shell. It strives to abstract backends to databases that are used to manage the workflow utilizing the different infrastructure and also the services. Switching for example to stage virtual machines from openstack clouds to amazon is as simple as specifying the name of the cloud. Moreover, cloudmesh client can be installed on Linux, MacOSX, and even Windows. Currently we support backends to SLURM, SSH, Openstack, Heat. In the past we supported AWS and Azure. We are in the process of integrating them back into the client.

This documentation and code is available on github at:

- Documentation:
  - on github: http://cloudmesh.github.io/client/
  - on rtd: http://cloudmesh-client.readthedocs.org/
- Code: https://github.com/cloudmesh/client
- Automated build reports and documentation:
  - Documentation: http://cloudmesh-client.readthedocs.org
  - Code: https://travis-ci.org/cloudmesh/client

CHAPTER 1

Prefix

Cloudmesh client is a simple client to enable access to multiple cloud environments form a command shell and commandline. It is grown out of the need to simplify access to multiple clouds for researchers and students easily. In contrast to our earlier versions of cloudmesh it explicitly separates the code to only target client code. Due to this simplification it is also possible to install the client code not only on Linux, OSX, but also Windows. We have tested the installation on Windows 10.

If you like to contribute or like to participate in the further development, please contact Gregor von Laszewski at laszewski@gmail.com.

1.1 Repositories

- Documentation: http://cloudmesh.github.io/client
- Code:
  - https://github.com/cloudmesh/base.git
  - https://github.com/cloudmesh/client.git
- Issues: https://github.com/cloudmesh/client/issues
- Milestones: https://github.com/cloudmesh/client/milestones
- Contributors: https://github.com/cloudmesh/client/graphs/contributors

As we have so far a tight integrated group, we are typically not forking the repository, but cloning it directly. Members are than able to work on the clones. We may change this in case we see need for forks.

1.1.1 Automated Builds and Reports

- Documentation: http://cloudmesh-client.readthedocs.org/
- Code: https://travis-ci.org/cloudmesh/client

1.2 Contact

For more info please contact Gregor von Laszewski, laszewski@gmail.com

Gregor von Laszewski E-mail: laszewski@gmail.comn Indiana University School of Informatics and Computing Informatics West 901 E. 10th St. Bloomington, IN 47408

and my office is at

611 N. Park Ave. Bloomington, IN 47408

Google Map

1.3 Authors

- 0001 (2015-06-13): Gregor von Laszewski (laszewski@gmail.com)
- 0002 (2015-06-15): fugangwang (kevinwangfg@gmail.com)
- 0003 (2015-06-15): Daniel Silva (silva10.daniel@gmail.com)
- 0004 (2015-06-18): Hyungro Lee (hroe.lee@gmail.com)
- 0005 (2015-06-18): Paulo Chagas (paulo.robertojr100@gmail.com)
- 0006 (2015-08-28): mangirish (vaglomangirish@gmail.com)
- 0007 (2015-08-28): Gourav Shenoy (shenoy.200@gmail.com)
- 0008 (2015-08-28): Mangirish Wagle (vaglomangirish@gmail.com)
- 0009 (2015-08-28): Erika Dsouza (erika27desouza@gmail.com)
- 0010 (2015-09-13): ehdsouza (Erika Dsouza)

1.4 Conventions

We will be using some simple conventions in this documentation. To indicate a command to be executed on the terminal we use *$* at the beginning of the line:

```
$ echo "Hello World"
```

A command started in the cloudmesh client shell is preceded by *cm*>:

```
cm> help
```

Often we are in the need to refer to a username or project. We will be using the username *albert* and the project id *FG-101*. It will be up to you to replace them with information related to your username and project. Alternatively we assume that you have set the shell variables $CM_USERNAME and $CM_PROJECT with for example:

```
$ export CM_USERNAME=albert
$ export CM_PROJECT=FG101
```

In this case we use in the documentation the values:

```
$CM_PROJECT
$CM_USERNAME
```

These values are typically set in the cloudmesh yaml file and if used they can be read from it into variables within cloudmesh scripts:

```
cm> var cloud=kilo
cm> var username=cloudmesh.profile.username
cm> var project=cloudmesh.clouds.$cloud.credentials.OS_TENANT_NAME
```

Please note that these values could be specific to a cloud as indicated by the example for the project in the above project is dependent on the specific cloud which can be easily integrated in the cloudmesh variables while using a *$* followed by the variable name.

1.5 Feature Requests

Please e-mail feature requests and bugs to laszewski@gmail.com.

We will manage them through github as part of issues and milestones:

- Issues: https://github.com/cloudmesh/client/issues
- Milestones: https://github.com/cloudmesh/client/milestones

Questions unrelated to cloudmesh but relate to futuresystems such as network issues and outages are best send through the form at

- https://portal.futuresystems.org/help

CHAPTER 2

Introduction

- 31 page slide presentation is available. Please press the download link. The presentation comes with audio and can only be utilized once you download it. An alternative link provides the presentation with audio online
- `8 page slide presentation`

Cloudmesh client allows to easily manage virtual machines, containers, HPC tasks, through a convenient client and API. Hence cloudmesh is not only a multi-cloud, but a multi-hpc environment that allows also to use container technologies (under development).

Client based. Cloudmesh client as the name indicates is a client based toolkit that is installed and run on the users computers. This also includes an add on component to the cloudmesh client which is a portal. Hence we distinguish the client that contains most of the functionality, as well as a portal that can access the functionality through a locally maintaine Web portal. Important to note is that the user manages its own credentials and thus security and credential management is done directly on the users machine instead through a hosted Web portal. This increases the security as access to any credential is managed by the user and is not part of a credential management system.

Layered Architecture. Cloudmesh client has a layered architecture that allows easy development of new features. This also allows contribution by the community while developing integrated and smaller sub components. Figure A depicts the various layers. A resource abstraction layer allows the integration of a multitude of resources spanning HPC, Containers, and Cloud resources. (At this time we focus on Openstack and Slurm resources. We are working on reintegrating resources such as Azure, AWS, Maui, Moab, and others which we previously supported, as well as new resources such as docker).

Fig. 2.1: Figure A: Cloudmesh layered architecture.

Management Framework. Cloudmesh client contains a management framework, and its components are depicted in Figure B. cloudmesh allows easy management of virtual machines, containers, and the data associated with them. We are currently developing a choreography framework that leverages Ansible, chef, and heat. All of the functionality is easily usable through a command shell that also can be used from the commandline, and a Python API. IN future we will be providing a REST API.

Fig. 2.2: Figure B: Cloudmesh component overview.

Database Agnostic. Cloudmesh contains some state about the resource and environment that a user may want to use. The information is managed in an database abstraction that would allow storing the data in a variety of databases such as SQL and MongoDB. At this time we have chosen SQLite to be the default database as it does not require any additional setup and is universally available on all operating systems without change.

Command shell and line. Cloudmesh contains a command shell allowing scripts to be developed and run. However we designed the command shell in such a way that each command can also be called from the command line. Through the cloudmesh state machine the state between command shell, command client, and the portal is shared.

Cloudmesh Client Portal. Previously, we distributed cloudmesh with client, server, and a portal components in one package. This however turned out to be to complex to be installed for some of our less technically skilled user community. Thus we split up the install into two independent packages. The cloudmesh client and the cloudmesh portal. The portal provides some elementary features to manage virtual machines and HPC jobs. At this time the portal is considered to be alpha technology. Just as the client the portal is to be run on the local user machine in oredr to allow increased security by managing the credentials locally rather than on a server.

Cloudmesh Two Factor Authentication. We have an exploratory project in place that looks at the use of Yubikeys for cloudmesh, client and cloudmesh portal.

Cloudmesh Comet. We are actively developing the client interface for SDSC's comet supercomputer allowing bare metal provisioning. The interface reuses cloudmesh components and technologies while interfacing with the comet cloud REST interface. The goal here is to manage virtual clusters.

2.1 Where to go next?

What to read next may depend on your interest. Certainly you want to install cloudmesh while following the

Installation information

Next we recommend that you get familiar with the concept of defaults in cloudmesh. After that you have several options:

- If you are interestted in clouds such as Openstack read the Section *Cloud Commands*
- If you are interested in Comet read the *the comet command manual*.
- If you are interested in HPC read the Section *HPC Commands*

CHAPTER 3

Quickstart

Warning: This quickstart quide assumes that you have prepared your system according to the steps documented in the Section *Preparation*.

3.1 Setup

Warning: The instalation with pip is not yet working at this time. Please use the instalation from source.

The setup of cloudmesh client is quite simple and can be done with:

```
pip install cloudmesh_client
```

However, you may want to read carefully our setup guide and prepare your machine as your OS may not have the required packages installed by default (see: *Preparation*).

3.2 Help

There are many commands in cloudmesh, and you can find out more about them while typing in

```
cm> help
```

When locationg a specific command you want to know more about, lets assume you want to know more about the command *color*, say

```
cm> help color
```

3.3 Understanding the cloudmesh shell

The cloudmesh shell contains a number of simple abstractions. This includes defaults, variables and configuration flags.

To set a default value, for example to set the default cloud to kilo use:

```
cm> default cloud=kilo
```

To configure color output of the cloudmesh shell use:

```
cm> color on
```

To conduct a live refresh in a cloud please use

```
cm> refresh on
```

3.4 Getting started with the Cloud(s)

Naturally you want to get started with clouds. In case you have a username and project in futuresystems using cloudmesh is easy. Only thing you need is an entry in the .ssh/config file with the machine name india, like follows:

```
Host india
Hostname india.futuresystems.org
User albert
```

Next you can register the cloud(s) with:

```
cm> register remote
```

This will fetch the necessary credentials from the cloud, and poplulate the cloudmesh.yaml file for you. At this time it will create an entry for a cloud named kilo.

If you need to view the flavors and images in the cloud, use:

```
cm> image refresh
cm> flavor refresh
```

To list the images/flavors use the following:

```
cm> list image
cm> list flavor
```

To set default flavor and image use:

```
cm> default image=Ubuntu 14.04
cm> default flavor=m1.tiny
```

You also need to set your default group. If you already have a group created you can use that or else you can specify a new group name.

```
cm> default group=test-group
```

Next, you need to upload your ssh keys to the cloud. If you already have a key-pair you can use it, or else you can generate ssh keys using:

```
$ ssh-keygen -t rsa -C albert@albert-pc
```

This will generate id_rsa.pub (public key) and id_rsa (private key) in the ~/.ssh/ directory.

First step (in the process of uploading key to cloud), is to add this key to the key database. To do so, use:

```
cm> key add --ssh --name=id_rsa
```

You can list the keys in the key database by using:

```
cm> key list
```

The output would look something like:

```
+--------+---------------+-------------------------------------+--------------+--------+
| name   | comment       | uri                                 | fingerprint  | source |
+--------+---------------+-------------------------------------+--------------+--------+
| id_rsa | albert@mycompi | file:///home/albert/.ssh/id_rsa.pub | 64:aa: ....  | ssh    |
+--------+---------------+-------------------------------------+--------------+--------+
```

Then, to upload this key to the cloud (your default cloud) use:

```
cm> key upload albert_ssh_key
```

3.5 Starting up a new VM in the cloud

If you have followed this document till this point, you are all set to start a new VM in the cloud. This section explains how to do that.

First, make sure all defaults are correctly set.

```
cm> vm default
```

The output will look somewhat similar to the following:

```
+-----------+---------------+
| Attribute | Value         |
+-----------+---------------+
| secgroup  |               |
| name      | albert-001    |
| image     | Ubuntu 14.04  |
| cloud     | kilo          |
| group     | test-group    |
| key       | id_rsa        |
| flavor    | m1.tiny       |
| login_key |               |
+-----------+---------------+
info. OK.
```

Starting a VM now is as simple as executing a single command.

```
cm> vm boot
```

This will start up a new VM in your default cloud. You need to refresh the database before listing VMs.

```
cm> vm refresh
cm> vm list
```

The output will look something like follows:

```
+----+--------------+------------+--------+-----------+-------------+----------+---------+--------+--
| id | uuid         | label      | status | static_ip | floating_ip | key_name | project | user   | c
+----+--------------+------------+--------+-----------+-------------+----------+---------+--------+--
| 47 | 8af4177f-... | albert-001 | ACTIVE | 10.0.2.37 |             | id_rsa   | fg478   | albert | k
+----+--------------+------------+--------+-----------+-------------+----------+---------+--------+--
```

Congratulations! you have now learnt how to set up cloudmesh, and use it to start a VM. Next step naturally is to login to the virtual machine. To do so, we need to assign it a public IP (also called floating IP).

To associate a floating ip to an instance (albert-001) in our case, use:

```
cm> network associate floating ip --instance=albert-001
```

Listing VMs will now show you this floating ip:

```
cm> vm list
```

```
+----+--------------+------------+--------+-----------+--------------+----------+---------+--------+
| id | uuid         | label      | status | static_ip | floating_ip  | key_name | project | user   |
+----+--------------+------------+--------+-----------+--------------+----------+---------+--------+
| 47 | 8af4177f-... | albert-001 | ACTIVE | 10.0.2.37 | 152.25.6.101 | id_rsa   | fg478   | albert |
+----+--------------+------------+--------+-----------+--------------+----------+---------+--------+
```

Next, you need to set your login key to be able to ssh to the VM. This will be the path to the private key (id_rsa) corresponding to the public key we uploaded to the cloud:

```
cm> default login_key=~/.ssh/id_rsa
```

Logging into the cloud is now as simple as:

```
cm> vm login albert-001
```

This should get you through to the ssh session to the VM. Congratulations! You have now learnt how to start a new VM and log into a VM.

To delete a VM, you use:

```
cm> vm delete albert-001
```

3.6 HPC

IN order to use the HPC experiment management functionality, you must register the queuing system in the yaml file and register the login node in the .ssh/config file. If you are using india and have used the clouds before, you may have already done this.

To start a command such as uname and execute a command you can say:

```
cm> run uname
```

It will print a job number that you may use to interact with the system further to for example list the output

```
cm> run list 101
```

(We assume here 101 is your job id)

To see the status and the output you can say

```
cm> run status 101
cm> run output 101
```

CHAPTER 4

Reference Card

4.1 Shell

Table 4.1: Shell

Command	Description
cm help	help
cm man	manual pages
cm script.cm	execute cm commands in script

4.2 Shell commands that expire after a session

Table 4.2: Shell

Command	Description
cm color on	sets the shell color
cm color off	switches off the color
cm refresh on	automatic refresh from the clouds
cm refresh off	data is only read from the database. Useful for managing thousands of VMs or limit your access to the cloud.
var a=xyx	declares a variable
var username=cloudmesh.profile.username	reads the variable from the cloudmesh.yaml file
var time=now	gets the time and store it in the variable time

4.3 Clouds

Table 4.3: Cloud

Command	Description
cm image list	list images
cm flavor list	list flavors
cm vm list	list vms
cm vm boot	boot vm
cm vm boot –cloud=kilo	boot vm on cloud kilo
cm default cloud=kilo	set default cloud to kilo
cm select image	select interactively the default image (not implemented yet).
cm select flavor	select interactively the default flavor (not implemented yet).
cm select cloud	select interactively the default cloud (not implemented yet).

4.4 HPC

Table 4.4: HPC

Command	Description
cm help	Help
cm hpc queue <batch>	info about the queue <batch>
cm hpc info	information about the queues on the HPC resource
cm hpc run uname -a	runs the command uname
cm hpc run list	prints the ids of previously run jobs
cm hpc run list	prints the ids of previously run jobs
cm hpc run list 11	prints the information regarding the job with the id 11

CHAPTER 5

Example Scripts

In this section we present a a number of scripts that may inspire you to utilize the scriting abilities of cloudmesh. A script can be started with

```
$ cm sriptname.cm
```

5.1 Comment

At this time comments are only detected based on the first characters in a line. A comment line starts either with #, // or /*.

comment.cm:

```
# test comment 1

// test comment 2

/* test comment 3 */
```

5.2 Terminal Commands

terminal.cm:

```
banner "prints a banner with this text"
echo "prints this text"
echo -r BLACK "prints in balck "
echo -r BLUE "prints in blue"
echo -r GREEN "prints in green"
```

5.3 Executing Shell commands

bash.cm:

```
# execute a shell command
! ls
```

5.4 Executing Python

py.cm:

```
py a = "hallo"
py print a

py n = 3
py for i in range(0,n): print i
```

5.5 Variables

var.cm:

```
#
# TESTING VARIABLES
#

# Listing the variables
var list

# Get the time and print a banner with the time
var a=now
banner $a

# get a variable from the cloudmesh yaml file and print it
var username=cloudmesh.profile.username
echo $username
```

5.6 Group

group.cm:

```
register remote kilo

# show teh defaukts
cm default
vm default

banner "GROUP A"
# boot 3 vms in goup a
default group=group_a

vm boot
vm boot
vm boot

vm list

banner "GROUP B"
# boot 3 vms in group b
default group=group_b
```

```
vm boot
vm boot
vm boot

vm list

# delete all vms in group_a
vm delete group_a

banner "GROUP C"
default group=group_c

# create three vms in group_c
py n = 3
py for i in range(0,n): cm vm boot

vm list

# Cleanup
vm delete group_b
vm delete group_c
```

5.7 Keys

key.cm:

```
banner KEYS

key delete --all

# TODO: does ~ expansion works?
key add --name=test-key ~/.ssh/id_rsa.pub

key list
key list --format=json
key list --format=yaml

key list --source=cloudmesh
key list --source=ssh
key list --source=git

key get test-key

default cloud
key add_to_cloud test-key

key list_cloud_mappings

key delete --select
```

5.8 VMs

vm.cm:

```
banner -r BLUE VM

banner -r BLUE -c "-" Setup

var cloud=kilo
var username=cloudmesh.profile.username
var tenant=cloudmesh.clouds.$cloud.credentials.OS_TENANT_NAME
var keyname="$username-key"

echo "Username: $username"
echo "Keyname:  $keyname"

register remote

default cloud=$cloud

default cloud

banner -r BLUE -c "-" "VM List"

vm refresh
vm list

key add --name $keyname ~/.ssh/id_rsa.pub
key list
key upload $keyname

default key=$keyname

default flavor=m1.small
default image=Ubuntu-14.04-64

#default flavor=2
#default image=9eb8416d-1313-4748-a832-5fe0ecbbdffc


default list --cloud=$cloud

vm default

vm boot

vm refresh

vm ip assign

vm list

vm status

default login_key=~/.ssh/id_rsa
# TODO: Monitor state change to check if the vm can be logged in
# vm login
```

```
# vm ssh uname -a

#--key=~/.ssh/id_rsa
=======
# according to scripts/secgroup.cm
# setting secgrup rule to allow ssh login
# secgroup rules-add --tenant=$tenant default 22 22 tcp 0.0.0.0/0
# however this seems having problem now

#default login_key=/home/mangirish/indiakey/id_rsa

#vm login --key=~/.ssh/id_rsa testvm

#vm list --format=json
#vm list --format=yaml
```

5.9 Copy

sync.cm:

```
default cloud=kilo

banner SYNC_FROM_LOCAL_TO_REMOTE
sync put ~/cm_sync/send sync_dir

banner SYNC_FROM_REMOTE_TO_LOCAL
sync get sync_dir/* ~/cm_sync/receive
```

5.10 Security Groups

secgroup.cm:

```
default cloud=kilo
var username=cloudmesh.profile.username
var tenant=cloudmesh.clouds.$cloud.credentials.OS_TENANT_NAME
var keyname="$username-key"

secgroup create --tenant=$tenant test-secgroup-01


banner LIST_SECURITY_GROUPS
secgroup list --tenant=$tenant


banner SECURITY_GROUP_ADD_RULES
secgroup rules-add --tenant=$tenant test-secgroup-01 80 80 tcp 0.0.0.0/0
secgroup rules-add --tenant=$tenant test-secgroup-01 443 443 udp 0.0.0.0/0


banner LIST_SECURITY_GROUP_RULES
secgroup rules-list --tenant=$tenant test-secgroup-01


banner SECURITY_GROUP_DELETE_RULE
```

```
secgroup rules-delete --tenant=$tenant test-secgroup-01 80 80 tcp 0.0.0.0/0
secgroup rules-list --tenant=$tenant test-secgroup-01


banner DELETE_SECURITY_GROUP
secgroup delete --tenant=$tenant test-secgroup-01


banner LIST_SECURITY_GROUPS
secgroup list --tenant=$tenant
```

5.11 Nova

nova.cm:

```
banner NOVA

# this command should be avoided and you should use the vm command instead.
# It is the same as ! nova ... but reads the configuration from the
# cloudmesh.yaml file

nova set kilo

nova info

nova list
nova image-list
```

5.12 Network

network.cm:

```
## lines that do not work but should are commented out for now with ##


banner SET_DEFAULT_CLOUD_AND_GROUP

var cloud=kilo
var username=cloudmesh.profile.username
var tenant=cloudmesh.clouds.$cloud.credentials.OS_TENANT_NAME
var keyname="$username-key"

echo "Username: $username"
echo "Keyname:  $keyname"

default group=demo_group


banner FLOATING_IP_LIST
network list floating ip


banner FLOATING_POOL_LIST
```

```
network list floating pool


banner CREATE_FLOATING_IP
network create floating
network list floating


banner LIST_VM_DEFAULTS
vm default


banner CREATE_VM
vm boot
vm refresh
vm list


banner ASSOCIATE_FLOATING_IP_AUTO_DETECT
network associate floating --instance=goshenoy


banner FLOATING_IP_LIST
network list floating ip


banner DISASSOCIATE_FLOATING_IP
network disassociate floating --instance=goshenoy


banner DELETE_GROUP
group delete demo_group


banner DEFAULT_GROUP
default group=demo_group


banner CREATE_VM
vm boot
vm refresh
vm list


banner ASSOCIATE_FLOATING_IP_WITH_GROUP
network associate floating --group=demo_group


banner DISASSOCIATE_FLOATING_IP_WITH_GROUP
network disassociate floating --group=demo_group


banner DELETE_GROUP
group delete demo_group
```

5.13 HPC

hpc.cm:

```
default cloud=general
default group=test
default cluster=india

banner HPC

hpc info

hpc queue

hpc run ~/test.sh --group=test1

hpc run ~/test.sh --group=test1

hpc status --job=6

hpc delete all --group=test1
```

5.14 Cluster

cluster.cm:

```
banner SET_DEFAULT_CLOUD_AND_GROUP

var cloud=kilo
var group=demo_group
var key=cloudmesh.keys.keylist.keyname
var username=cloudmesh.profile.username

default cloud=$cloud
default group=$group
default key=$key

banner LIST_VM_DEFAULTS
vm default

banner CREATE_VM
vm boot
vm refresh
vm boot
vm refresh
vm list

banner CREATE_VIRTUAL_CLUSTER
network create cluster --group=$group
```

5.15 Cloud

cloud.cm:

```
banner DEFAULTS
list default


banner LIST_CLOUDS
cloud list


banner LOGON_TO_A_CLOUD
cloud logon kilo
cloud list


banner DEACTIVATE_A_CLOUD
cloud deactivate kilo
cloud list


banner ACTIVATE_A_CLOUD
cloud activate kilo
cloud list


banner LOGOUT_FROM_CLOUD
cloud logout kilo
cloud list
```

CHAPTER 6

Setup

6.1 Preparation

The installation of cloudmesh is easy if you have prepared your system with up-to-date software. We provide instructions to prepare your system for a number of operating systems. After you have completed the system preparation you can follow the Installation instructions which will be the same for all systems.

In future we will provide an even simpler install mechanism on the various operating systems based on simple install scripts.

If you like to help us in making the instructions simpler based on your experience, please email us or create a pull request in github.

6.1.1 OSX

You will need a number of tools that are not distributed with the regular OSX operating system. First you need to install xcode. The easiest is to open a terminal and type

```
$ xcode-select --install
```

We recommend that you use python 2.7, e.g. at least python 2.7.10. This version of python is easy to install while downloading the dmg and installing it on the system. You can find the python version at:

- https://www.python.org/downloads/

You will still have access to the python version distributed with the original OSX operating system. You will need to install pip, and virtualenv which you can do with

```
$ sudo easy_install pip
$ sudo pip install virtualenv
```

To test out which version you have activated, you can use in the command line

```
$ python --version
$ pip --version
$ virtualenv --version
```

Make sure that you have the supported versions:

Software	Version
Python	2.7.10
pip	8.0.2
virtualenv	13.1.2

On OSX as well as the other operating systems we **require** you to use virtualenv. First you need to find which version of python you use. You can say

```
$ which python
```

It will give you the path of the python interpreter. Let us assume the interpreter was found in */usr/local/bin/python*. Next you can create a virtual ENV with

```
$ virtualenv -p /user/local/bin/python ~/ENV
```

You will need to activate the virtualenv with

```
$ source ~/ENV/bin/activate
$ export PYTHONPATH=~/ENV/lib/python2.7/site-packages:$PYTHONPATH
```

If successful, your terminal will have (ENV) as prefix to the prompt:

```
(ENV)machinename:dirname user$
```

If you like to use this version of python consistently, you may elect to add it to your .bashrc file and add the command:

```
source $HOME/ENV/bin/activate
export PYTHONPATH=~/ENV/lib/python2.7/site-packages:$PYTHONPATH
```

We need to just do some simple updates in the virtualenv and you will have an up to date python environment in ~/ENV

```
$ pip install pip -U
$ easy_install readline
$ easy_install pycrypto
$ pip install urllib3
```

Warning: We found that `readline` and `pycrypto` could not be installed with pip at the time of writing of this manual, despite the fact that pip claimed to have installed them. However, the version installed with pip were not usable. The workaround is to use easy_install for these packages as shown above. If you have better idea how to fix this, let us know and send mail to laszewski@gmail.com.

It is recommended that you test the version of the python interpreter and pip again

```
$ pip --version
```

which should give the version 8.0.2

```
$ python --version
```

which should give the version Python 2.7.10

OSX Quick Install Scripts (untested)

Use at your own risk, we recommend that you follow the more detailed instructions above

```
$ xcode-select --install
$ open https://www.python.org/downloads/
```

Install python 2.7.10. Next do

```
$ sudo easy_install pip
$ sudo pip install virtualenv
$ virtualenv -p /user/local/bin/python ~/ENV
```

```
$ source ~/ENV/bin/activate
$ export PYTHONPATH=~/ENV/lib/python2.7/site-packages:$PYTHONPATH
$ pip install pip -U
$ easy_install readline
$ easy_install pycrypto
$ pip install urllib3
```

In case you have not added the two lines in your .bashrc script, you will need to run them in any new terminal you start in which yo like to use the new python version. It may just be easier to add them to your .bashrc file.

source ~/ENV/bin/activate export PYTHONPATH=~/ENV/lib/python2.7/site-packages:$PYTHONPATH

6.1.2 Ubuntu 14.04/15.04

As your ubuntu version may be outdated we ask you to run the following commands

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get dist-upgrade
$ sudo apt-get install python-setuptools
$ sudo apt-get install python-pip
$ sudo apt-get install python-dev
$ sudo apt-get install libncurses-dev
$ sudo apt-get install git
$ sudo easy_install readline
$ sudo pip install pycrypto
$ sudo apt-get install build-essential checkinstall
$ sudo apt-get install libreadline-gplv2-dev
$ sudo apt-get install libncursesw5-dev
$ sudo apt-get install libssl-dev
$ sudo apt-get install libsqlite3-dev
$ sudo apt-get install tk-dev
$ sudo apt-get install libgdbm-dev
$ sudo apt-get install libc6-dev
$ sudo apt-get install libbz2-dev
```

Note: if pycrypto does not install with pip use easy_install pycrypto

We recommend that you use python 2.7.10, which you can install it alternatively in your system with without overwriting the existing python version

```
$ cd $HOME
$ wget --no-check-certificate https://www.python.org/ftp/python/2.7.10/Python-2.7.10.tgz
$ wget --no-check-certificate https://bitbucket.org/pypa/setuptools/raw/bootstrap/ez_setup.py
$ wget --no-check-certificate https://bootstrap.pypa.io/get-pip.py
$ tar xzf Python-2.7.10.tgz
$ cd Python-2.7.10
$ ./configure --prefix=/usr/local
$ sudo make && sudo make altinstall
$ export PATH="/usr/local/bin:$PATH"
```

Verify if you now have the correct alternative python installed

```
$ /usr/local/bin/python2.7 --version
```

which will return Python 2.7.10. Next, Install setuptools and pip

```
$ cd $HOME
$ sudo /usr/local/bin/python2.7 ez_setup.py
$ sudo /usr/local/bin/python2.7 get-pip.py
```

Create soft symbolic links

```
$ sudo ln -sf /usr/local/bin/python2.7 /usr/local/bin/python
$ sudo ln -sf /usr/local/bin/pip /usr/bin/pip
```

Verify if you now have the required pip version installed

```
$ pip --version
```

It shoudl show the version 8.0.2. If you see a lower version of pip, you may upgrade it with the following command

```
$ pip install -U pip
```

Next, Install a python virtual environment on your machine as we do not want to interfere with the system installed python versions. Inside your terminal run

```
$ sudo apt-get install virtualenv
```

Next we will create a python virtualenv in the directory $HOME/ENV. To activate virtualenv, execute the following steps

```
$ virtualenv -p /usr/local/bin/python $HOME/ENV
$ source $HOME/ENV/bin/activate
```

This will add a '(ENV)' to your prompt in the terminal like following:

```
(ENV)[user@hostname ~]$
```

Ubuntu Quick Install Scripts (untested)

Use at your own risk, we recommend that you follow the more detailed instructions above. THe script bellow contains also an update of the python version from 2.7.9 to 2.7.10 in an alternate install. As cloudmesh is running fine in python 2.7.9 the update may not be needed and you may eliminate the steps in regards to this from the bellow script if you wish.

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get dist-upgrade
$ sudo apt-get install python-setuptools
$ sudo apt-get install python-pip
$ sudo apt-get install python-dev
$ sudo apt-get install libncurses-dev
$ sudo apt-get install git
$ sudo easy_install readline
$ sudo pip install pycrypto
$ sudo apt-get install build-essential checkinstall
$ sudo apt-get install libreadline-gplv2-dev
$ sudo apt-get install libncursesw5-dev
$ sudo apt-get install libssl-dev
$ sudo apt-get install libsqlite3-dev
$ sudo apt-get install tk-dev
$ sudo apt-get install libgdbm-dev
$ sudo apt-get install libc6-dev
$ sudo apt-get install libbz2-dev
```

```
$ cd $HOME
$ wget --no-check-certificate https://www.python.org/ftp/python/2.7.10/Python-2.7.10.tgz
$ wget --no-check-certificate https://bitbucket.org/pypa/setuptools/raw/bootstrap/ez_setup.py
$ wget --no-check-certificate https://bootstrap.pypa.io/get-pip.py
$ tar xzf Python-2.7.10.tgz
$ cd Python-2.7.10
$ ./configure --prefix=/usr/local
$ sudo make && sudo make altinstall
$ export PATH="/usr/local/bin:$PATH"
$ cd $HOME
$ sudo /usr/local/bin/python2.7 ez_setup.py
$ sudo /usr/local/bin/python2.7 get-pip.py
$ sudo ln -sf /usr/local/bin/python2.7 /usr/local/bin/python
$ sudo ln -sf /usr/local/bin/pip /usr/bin/pip
$ pip install -U pip
$ virtualenv -p /usr/local/bin/python $HOME/ENV
```

Add the following to your .bashrc file:

```
source $HOME/ENV/bin/activate
```

6.1.3 CentOS

This documentation assumes that the user is advanced enough to use linux terminal. We also assume you are not logged in as root, but you are a regular user. However to prepare the system we assume you have sudo privileges.

One line install

You can conduct these steps automatically as well as the install of cloudmesh by executing the following script in your command line.

After this you not only have the system updated for coudmesh with necessary libraries and tools, but you will also have cloudmesh installed.

We encourage you to inspect the script and assess if this is the way you like to proceed. If you rather do a step by step install, please read on.

Deatailed Step-by-Step system preparation

I you like to conduct these steps by hand please read on. First, we check for up-to-date versions of python and pip

```
$ python --version
```

As CentOS typically comes with an old version of python (2.7.5), we will install in addition to the system provided python, an alternative python installation. This is achieved by following the next steps executing them as normal user. They will install python 2.7.10 under‘$HOME/ENV‘

```
$ sudo yum install -y gcc wget zlib-devel openssl-devel sqlite-devel bzip2-devel
$ cd $HOME
$ wget --no-check-certificate https://www.python.org/ftp/python/2.7.10/Python-2.7.10.tgz
$ wget --no-check-certificate https://bitbucket.org/pypa/setuptools/raw/bootstrap/ez_setup.py
$ wget --no-check-certificate https://bootstrap.pypa.io/get-pip.py
$ tar -xvzf Python-2.7.10.tgz
$ cd Python-2.7.10
$ ./configure --prefix=/usr/local
```

```
$ sudo make && sudo make altinstall
$ export PATH="/usr/local/bin:$PATH"
```

Verify if you now have the correct alternative python installed

```
$ /usr/local/bin/python2.7 --version
```

which should return Python 2.7.10. Next, install setuptools and pip and create symbolic links to them

```
$ cd $HOME
$ sudo /usr/local/bin/python2.7 ez_setup.py
$ sudo /usr/local/bin/python2.7 get-pip.py
$ sudo ln -s /usr/local/bin/python2.7 /usr/local/bin/python
$ sudo ln -s /usr/local/bin/pip /usr/bin/pip
```

Verify if you now have the required pip version installed (this may require a new terminal to test or a source or the .bashrc script)

```
$ pip --version
$ pip 8.0.2 from /usr/lib/python2.7/site-packages/pip-8.0.2-py2.7.egg (python 2.7)
```

If you see an older version of pip, upgrade it with the following command

```
$ pip install -U pip
```

Next, Install a python virtual environment on your machine as we do not want to interfere with the system installed python versions. Inside your terminal run

```
$ sudo pip install virtualenv
```

Next we will create a python virtualenv in the directory $HOME/ENV. To activate virtualenv, execute the following steps

```
$ virtualenv -p /usr/local/bin/python $HOME/ENV
$ source $HOME/ENV/bin/activate
```

This will add a '(ENV)' to your prompt in the terminal like following:

```
(ENV)[user@hostname ~]$
```

On more permanent basis, if you want to avoid activating virtualenv every time you log in, You can add the activation of the virtualenv to the ~/.bashrc file with your favourate editor:

```
emacs ~/.bashrc
```

Add the command:

```
source $HOME/ENV/bin/activate
```

to the file and save the file. You may test if this works, by launching a new terminal session and checking if (ENV) is seen added to the prompt.

Centos Quick Install Scripts

Use at your own risk, we recommend that you follow the more detailed instructions above

```
$ sudo yum install -y gcc wget zlib-devel openssl-devel sqlite-devel bzip2-devel
$ cd $HOME
$ wget --no-check-certificate https://www.python.org/ftp/python/2.7.10/Python-2.7.10.tgz
```

```
$ wget --no-check-certificate https://bitbucket.org/pypa/setuptools/raw/bootstrap/ez_setup.py
$ wget --no-check-certificate https://bootstrap.pypa.io/get-pip.py
$ tar -xvzf Python-2.7.10.tgz
$ cd Python-2.7.10
$ ./configure --prefix=/usr/local
$ sudo make && sudo make altinstall
$ export PATH="/usr/local/bin:$PATH"
$ cd $HOME
$ sudo /usr/local/bin/python2.7 ez_setup.py
$ sudo /usr/local/bin/python2.7 get-pip.py
$ sudo ln -s /usr/local/bin/python2.7 /usr/local/bin/python
$ sudo ln -s /usr/local/bin/pip /usr/bin/pip
$ pip install -U pip
$ sudo pip install virtualenv
$ virtualenv -p /usr/local/bin/python $HOME/ENV
```

Add the following to your .bashrc script:

```
source $HOME/ENV/bin/activate
```

6.1.4 Windows 10

Install Python

Python can be found at http://www.python.org. We recommend to download and install the newest version of python. At this time we recommend that you use version 2.7.10. Other versions may work to, but are not supported or tested. A direct link to the install can be found at:

```
https://www.python.org/ftp/python/2.7.10/python-2.7.10.msi
```

In powershell you need to type:

```
PS> explorer https://www.python.org/ftp/python/2.7.10/python-2.7.10.msi
```

This will open the internet browser and download the python msi installer. It will walk you through the install process.

Note: If you like to install it separately, you can find the downloaded msi in the *~/Downloads* directory. To install it in powershell use:

```
PS> cd ~/Downloads
PS> msiexec /i python-2.7.10.msi /qb
```

This will open a basic dialog to perform installation and close after completion.

Note: While installing python, you have the option to automatically include python binaries in the system Path. This is disabled by default, so you will need to enable it explicitly. Skip below step if you have choose to enable this feature.

After you have installed python (and not explicitly enabled the feature to add python to system path) include it in the Path environment variable while you type in powershell:

```
PS> [Environment]::SetEnvironmentVariable("Path", "$env:Path;C:\Python27\;C:\Python27\Scripts\", "Use
PS> $env:Path=[Environment]::GetEnvironmentVariable("Path", "User")
```

This should install Python 2.7.10 successfully. You can now proceed to the next step.

Install Chocolatey, Git, VirtualEnv, Make

As we need to do some editing you will need a nice editor. Please do not use notepad and notepad++ as they have significant issues, please use vi, vim, or emacs. Emacs is easy to use as it has a GUI on windows. Install emacs:

```
PS> Start-Process powershell -Verb runAs
```

This will open a new Powershell window with administrator privileges. Continue the below steps to install chocolatey & make:

```
PS> Set-ExecutionPolicy Unrestricted -force
PS> iex ((new-object net.webclient).DownloadString('https://chocolatey.org/install.ps1'))
PS> choco install emacs -y
PS> choco install make -y
```

Next, to install Git, type the following command into powershell:

```
PS> explorer https://git-scm.com/download/win
```

This will open the internet browser and download the git installer. It will walk you through the install process.

Note: When installing you will see at one point a screen that asks you if you like to add the commands to the shell. It is recommended you select option (3) to add Unix shell commands to windows. This will install Unix style commands to Windows and include it in path.

Follow the on screen instructions, selecting the default values for all of the options (except for above note). This will install Git & Git Bash successfully.

Install VirtualEnv and Create a Virtual Python Environment

At the time this guide was written, the latest version of python virtualenv was 14.0.2. But Windows 10 users were facing a lot of issues with this version, and so we recommend installing a lower version of virtualenv:

```
PS> pip install virtualenv==13.0.2
```

This will install python virtualenv on your system. To setup the environment in powershell, run the following command:

```
PS> virtualenv ~/ENV
```

This will create a new directory *~/ENV/* comprising a local python environment. To activate this new environment, run:

```
PS> ~/ENV/Scripts/activate.ps1
```

This will activate your new python virtual environment. As a proof, you will now see a *(ENV)* prefixed to the powershell. It will look like:

```
(ENV) PS> python --version
          Python 2.7.10
```

Congratulations, you have now activated your python virtualenv.

Note: To deactivate this virtualenv, you need to run the following command:

```
(ENV) PS> deactivate
```

But always remember to activate the virtualenv before using cloudmesh.

Next step is to install necessary python packages.

Install Pycrypto

First, if not already done, activate your virtualenv:

```
PS> ~/ENV/Scripts/activate.ps1
```

Next, update your python-pip:

```
(ENV) PS> pip install pip -U
```

Check the python and pip version:

```
(ENV) PS> python --version
      Python 2.7.10

(ENV) PS> pip --version
      pip 8.0.2 from c:\users\test-pc\ENV\lib\site-packages (python 2.7)
```

Then to install pycrypto, run the following:

```
(ENV) PS> easy_install http://www.voidspace.org.uk/python/pycrypto-2.6.1/pycrypto-2.6.1.win32-py2.7.e
```

Install FireFox Browser

Cloudmesh contains tools for generating and viewing the html documentation files. It uses FireFox to render HTML pages. To install FireFox, run the following command:

```
(ENV) PS> explorer https://www.mozilla.org/en-US/firefox/new/#download-fx
```

This will download the latest FireFox browser installer on your machine. Follow the on-screen instructions to install. Once complete, add FireFox to your path:

```
(ENV) PS> [Environment]::SetEnvironmentVariable("Path", "$env:Path;C:\Program Files (x86)\Mozilla Fi
(ENV) PS> $env:Path=[Environment]::GetEnvironmentVariable("Path", "User")
```

COngratulations! You have now successfully setup your Windows 10 machine, and are all ready to now install Cloudmesh.

Adding SSH Key to Futuresystems Portal

Close the current Powershell window and open a new one. Now we are ready to use ssh and git. But first, let's create a key:

```
PS> ssh-keygen -t rsa
```

Follow the instructions and leave the path unchanged. Make sure you specify a passphrase. It is a policy on many compute resources that your key has a passphrase. Look at the public key as we will need to upload it to some resources:

```
PS> cat ~/.ssh/id_rsa.pub
```

Go to the futuresystems portal:

```
https://portal.futuresystems.org
```

Once you log in you can use the following link to add your public key to futuresystems:

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

Naturally this only works if you are eligible to register and get an account. Once you are in a valid project you can use indias resources. After that you need to upload your public key that you generated into the portal and did a cat on.

Warning: Windows will not past and copy correctly, please make sure that newlines are removed for the text box where you past the key. This is cause for many errors. Make sure that the key in the text box is a single line and looks like when you did the cat on it.

To simplify SSH access, you will need to configure a ssh config file. You will need to first create a *config* file as follows:

```
PS> vim ~/.ssh/config
```

This should open the VIM editor and next you need to enter the following contents:

```
Host india
     Hostname india.futuresystems.org
     User <your_portal_username>
     IdentityFile <path_to_id_rsa_file>
```

Replace *your_portal_username* with your futuresystems username and *path_to_id_rsa_file* with the path to your private key file. It generally is at ~/.ssh/id_rsa.

You can now easily perform ssh to futuresystems cloud using:

```
PS> ssh india
```

6.2 Installation

We assume that you have prepared your system (see Section _my-reference-label) on which you like to install the cloudmesh client. We recommend that you use python 2.7.10, pip 7.1.2 and use virtualenv. Furthermore we recommend that on Linux systems you have readline installed as it is a convenient tool for command line manipulation. In the next sections we will walk you through a setup that has been proven to work for developers and users and is very easy to replicate.

6.2.1 Install Cloudmesh Client via pip

Warning: The instalation with pip is not yet working at this time. Please use the instalation from source

Users can install the cloudmesh client via pip

```
$ pip install cloudmesh_client
```

6.2.2 Cloudmesh Installation from Source

Developers that wish to contribute to the source can obtain the code from github. We assume that we conduct a source code install into the directory:

```
~/github/cloudmesh
```

If you like to use a different directory, that is also possible, but the instructions we provide here assumes are targeted towards this base directory.

Please use the following commands

```
$ mkdir -p github/cloudmesh
$ cd github/cloudmesh
$ git clone https://github.com/cloudmesh/base.git
$ git clone https://github.com/cloudmesh/client.git
$ cd base
$ python setup.py install
$ cd ../client
$ python setup.py install
```

Updating an existing source distribution

During the development phase of cloudmesh you may need to update the code from source, as cloudmesh client uses three different repositories please do not forget to update them accordingly

```
$ export CLOUDMESH_HOME=$HOME/github/cloudmesh
$ cd $CLOUDMESH_HOME/base
$ git pull
$ python setup.py install
$ cd $CLOUDMESH_HOME/client
$ python setup.py install
```

Testing

Todo

This section is incomplete and we need to make sure that tox works. We also need to explain how travis works and how we can run nosetests locally

For now we do not assume that you need to run any tests after you install the source. We will address deployment tests later.

```
$ pip install tox
```

in the source dir say

```
$ tox
```

Nose tests can be started with

```
$ nosetests
```

6.3 Configuration

During the installation of cloudmesh it will automatically generate a configuration file in the directory:

~/.cloudmesh/cloudmesh.yaml

If this file is missing, you can run the command:

cm help to automatically generate it from defaults.

The file will be a template and it can either be modified with your favorite editor, or if you are at Indiana University and want to use the kilo cloud you can use the command

cm remote register

This will add the appropriate information into the yaml file. The file will be looking as follows. You will have several options to modify the file as explained bellow

```
meta:
    yaml_version: 3.0
    kind: clouds
cloudmesh:
    profile:
        firstname: TBD
        lastname: TBD
        email: TBD
        username: None
    github:
        username: TBD
    portal:
        location: TBD
        browser: firefox
    comet:
        auth_provider: userpass
        userpass:
            username: TBD
            password: TBD
        apikey:
            api_key: TBD
            api_secret: TBD
    hpc:
        experiment:
           name: gregor-00000
        active:
        - comet
        - juliet
        clusters:
            india:
                cm_heading: India HPC CLuster
                cm_host: india
                cm_label: indiahpc
                cm_type: slurm
                cm_type_version: 14.11.9
                credentials:
                    username: TBD
                    project: None
                default:
                    queue: delta
                    experiment_dir: /N/u/{username}/experiment
                    prefix: {username}
```

```
        comet:
            cm_heading: Comet HPC CLuster
            cm_host: comet
            cm_label: comethpc
            cm_type: slurm
            cm_type_version: 14.11.9
            credentials:
                username: TBD
                project: None
            default:
                queue: debug
                experiment_dir: /home/{username}/experiment
                prefix: {username}
active:
    - kilo
clouds:
    kilo:
        cm_heading: India OpenStack, Kilo
        cm_host: india
        cm_label: kilo
        cm_type: openstack
        cm_type_version: kilo
        cm_openrc: ~/.cloudmesh/clouds/india/kilo/openrc.sh
        credentials:
            OS_AUTH_URL: https://kilo.futuresystems.org:5000/v3
            OS_PASSWORD: TBD
            OS_TENANT_NAME: TBD
            OS_USERNAME: TBD
            OS_PROJECT_DOMAIN_ID: default
            OS_USER_DOMAIN_ID: default
            OS_PROJECT_NAME: TBD
            OS_IMAGE_API_VERSION: 2
            OS_VOLUME_API_VERSION: 2
        default:
            flavor: TBD
            image: TBD
    chameleon:
        cm_heading: Chameleon
        cm_host: chameleoncloud.org
        cm_label: chameleon
        cm_type: openstack
        cm_type_version: kilo
        credentials:
            OS_AUTH_URL: https://openstack.tacc.chameleoncloud.org:5000/v2.0
            OS_PASSWORD: TBD
            OS_TENANT_NAME: TBD
            OS_TENANT_ID: TBD
            OS_PROJECT_NAME: TBD
            OS_USERNAME: TBD
            OS_VERSION: kilo
            OS_REGION_NAME: RegionOne
        default:
            flavor: m1.small
            image: Ubuntu-Server-14.04-LTS
    cybera-c:
      cm_heading: Cybera Calgary OpenStack
      cm_host: cybera
      cm_label: cybera-c
```

```
      cm_type: openstack
      cm_type_version: kilo
      credentials:
        OS_AUTH_URL: TBD
        OS_TENANT_ID: TBD
        OS_TENANT_NAME: TBD
        OS_PROJECT_NAME: TBD
        OS_USERNAME: TBD
        OS_PASSWORD: TBD
        OS_REGION_NAME: Calgary
      default:
        flavor: m1.small
        image: Ubuntu 14.04
    cybera-e:
      cm_heading: Cybera Edmonton OpenStack
      cm_host: cybera
      cm_label: kilo
      cm_type: openstack
      cm_type_version: kilo
      credentials:
        OS_AUTH_URL: https://keystone-yyc.cloud.cybera.ca:5000/v2.0
        OS_TENANT_ID: TBD
        OS_TENANT_NAME: TBD
        OS_PROJECT_NAME: TBD
        OS_USERNAME: TBD
        OS_PASSWORD: TBD
        OS_REGION_NAME: Edmonton
      default:
        flavor: m1.small
        image: Ubuntu 14.04
    aws:
        cm_heading: Amazon Cloud, AWS
        cm_host: aws.amazon.com
        cm_label: aws
        cm_type: ec2
        cm_type_version: null
        credentials:
            EC2_ACCESS_KEY: TBD
            EC2_SECRET_KEY: TBD
            keyname: TBD
            userid: TBD
        default:
            flavor: t1.micro
            image: ami-d85e75b0
            location: us-east-1
    chameleon-ec2:
        cm_heading: Chameleon, EC2
        cm_host: chameleoncloud.org
        cm_label: chameleon_ec2
        cm_type: ec2
        cm_type_version: ec2
        credentials:
            EC2_ACCESS_KEY: TBD
            EC2_SECRET_KEY: TBD
            keyname: TBD_not_used
            userid: TBD_not_used
            EC2_URL: https://openstack.tacc.chameleoncloud.org:8773/services/Cloud
            EC2_USER_ID: TBD
```

```
                EC2_PRIVATE_KEY: ~/.cloudmesh/clouds/chameleon/TBD/pk.pem
                EC2_CERT: ~/.cloudmesh/clouds/chameleon/TBD/cert.pem
                NOVA_CERT: ~/.cloudmesh/clouds/chameleon/TBD/cacert.pem
                EUCALYPTUS_CERT: ~/.cloudmesh/clouds/chameleon/TBD/cacert.pem
            default:
                flavor: m1.small
                image: Ubuntu-Server-14.04-LTS
        azure:
            cm_heading: Microsoft Azure Virtual Machines
            cm_host: windowsazure.com
            cm_label: azure
            cm_type: azure
            cm_type_version: null
            credentials:
                managementcertfile: TBD
                servicecertfile: TBD
                subscriptionid: TBD
                thumbprint: TBD
            default:
                flavor: ExtraSmall
                image: b39f27a8b8c64d52b05eac6a62ebad85__Ubuntu-14_04_2-LTS-amd64-server-20150610-en-
                location: East US
    keys:
        default:
        keylist:
          keyname: ~/.ssh/id_rsa.pub
    system:
        data: ~/.cloudmesh/cloudmesh_inventory.yaml
        console_color: true
    logging:
        file: ~/.cloudmesh/cloudmesh.log
        level: DEBUG
```

You can modify the file by hand and replace the `TBD` values according to your information about your cloud. You can add new clouds or delete the once that you do not want.

Warning: Just as private keys should be kept private so does the cloudmesh.yaml file. Please, make sure the file is protected as it contains
sensitive information.

6.3.1 Get Registration from Indiana University

In case you have an account on http::/portal.futuresystems.org the integration can be done automatically for you with the account information available to you as previously explained.

The best way is to configure first your ssh client to conveniently log into india the machine where you can find the configuration information. To do so, please edit the following file

```
~/./ssh/config
```

and add the following lines to it

```
Host india
    User: albert
    Hostname: india.futuresystems.org
```

please replace albert with your portal name that you have used for registration with futuresystems.org. Once you have done this please verify that you have access to india with a command such as:

```
ssh india uname -a
```

Next register the FutureSystems clouds into your cloudmesh yaml file with the command:

```
cm register remote
```

This will update your cloudmesh.yaml file with the information retrieved from india. While retrieving the information on india from the file:

```
~/.cloudmesh/clouds/india/kilo/openrc.sh
```

Make sure you add a valid tenant to the yaml file. More information about using india can be found at http://portal.futuresystems.org

6.3.2 Registration of other clouds

The register command is quite powerful and useful and we encourage you to take a closer look at the manual pages. This includes command such as

To find out more about the registration command:

```
cm register help
```

To edit the yaml file with the editor defined by the Shell variable *$EDITOR*:

```
register edit
```

To list the *cloudmesh.yaml* file:

```
register list
```

6.3.3 Registration of Cybera Cloud

Cybera an organization from Canada provides an easy accessible openstack cloud. This cloud should only be used while following their access policies documented at:

- http://www.cybera.ca/projects/cloud-resources/rapid-access-cloud/faq/#What_is_RAC

YOu may ask for permission, if you do not fit this category. Once you have created an account at:

- https://rac-portal.cybera.ca/

YOu can access to Openstack portal at

- https://cloud.cybera.ca/auth/login/

Just as Chameleon Cloud the Cybera cloud allows openstack rc and ec2 rc files.

Registration of Cybera Openstack Cloud

When you have an account ana a project it is simple to configure cloudmesh to include chameleon cloud in its resource set. To do so, edit the file:

~/.cloudmesh/cloudmesh.yaml

Edit the follwoing lines:

```
OS_PASSWORD: TBD
OS_TENANT_NAME: TBD
OS_TENANT_ID: TBD
OS_PROJECT_NAME: TBD
OS_USERNAME: TBD
```

Let us assume you have the username *albert* and the project id *FG-101*, Than the lines need to be changed to:

```
OS_PASSWORD: <your user password>
OS_TENANT_NAME: FG-101
OS_TENANT_ID: FG-101
OS_PROJECT_NAME: FG-101
OS_USERNAME: albert
```

You can find this information also in the openrc.sh file which you can download via the Openstack Horizon interface by following this link:

- https://cloud.cybera.ca/project/access_and_security/api_access/openrc/

Registration of Cybera EC2 Cloud

Cybera cloud also support the usage of the EC2 interface which is a pit more complex to set up than the openstack configuration. First, you have to download a configuration directory, that is packaged as a zip file. This file can be found at

- https://cloud.cybera.ca/project/access_and_security/api_access/ec2/

Let us assume you have the username albert and the project FG-101. Than the zip file will be called:

```
FG-101-x509.zip
```

Let us set some environment variables to make the configuration description easier

```
$ export C_USERNAME=<your cybera username>
$ export C_PROJECT=<your cybera project name>
```

Unpack the zip file and place the entire directory in the .cloudmesh directory with. (We assume that you are in the directory where your browser downloaded the zip file and you have uncompressed it)

```
$ mkdir ~/.cloudmesh/clouds/cybera/$C_PROJECT
$ cp $C_PROJECT ~/.cloudmesh/clouds/cybera/$C_PROJECT
$ ls ~/.cloudmesh/clouds/cybera/$C_PROJECT
```

The directory should include the files:

```
cacert.pem
cert.pem
ec2rc.sh
pk.pem
```

Take a look at the ec2rc.sh file

```
$ cat ~/.cloudmesh/clouds/cybera/$C_PROJECT/ec2rc.sh
```

Now you can edit the cloudmesh yaml file at:

```
~/.cloudmesh/cloudmesh.yaml
```

locate the cybera-ec2 entry and change the TBD values with the values you see in the ec2rc.sh file:

```
EC2_ACCESS_KEY: <find the value in the ec2rc.sh file>
EC2_SECRET_KEY: <find the value in the ec2rc.sh file>
EC2_USER_ID: <find the value in the ec2rc.sh file>
```

For the following lines in the cloudmesh file, please replace the TBD values with the cybera project ID that you use for this cloud:

```
EC2_PRIVATE_KEY: ~/.cloudmesh/clouds/cybera/TBD/pk.pem
EC2_CERT: ~/.cloudmesh/clouds/cybera/TBD/cert.pem
NOVA_CERT: ~/.cloudmesh/clouds/cybera/TBD/cacert.pem
EUCALYPTUS_CERT: ~/.cloudmesh/clouds/cybera/TBD/cacert.pem
```

6.3.4 Chameleon Cloud

Registration of Chameleon Openstack Cloud

NSF sponsors an experimental cloud environment called Chameleon at

- https://www.chameleoncloud.org

It is a KVM based Openstack cloud of version kilo. The documentation can be found here:

- https://www.chameleoncloud.org/docs/user-guides/openstack-kvm-user-guide/

When you have an account and a project it is simple to configure cloudmesh to include chameleon cloud in its resource set. To do so, edit the file:

~/.cloudmesh/cloudmesh.yaml

Edit the follwoing lines:

```
OS_PASSWORD: TBD
OS_TENANT_NAME: TBD
OS_TENANT_ID: TBD
OS_PROJECT_NAME: TBD
OS_USERNAME: TBD
```

Let us assume you have the username *albert* and the project id *FG-101*, Than the lines need to be changed to:

```
OS_PASSWORD: <your user password>
OS_TENANT_NAME: FG-101
OS_TENANT_ID: FG-101
OS_PROJECT_NAME: FG-101
OS_USERNAME: albert
```

You can find this information also in the openrc.sh file which you can download via the Openstack Horizon interface by following this link:

- https://openstack.tacc.chameleoncloud.org/dashboard/project/access_and_security/api_access/openrc/

Registration of Chameleon EC2 Cloud

The chameleon cloud also support the usage of the EC2 interface which is a pit more complex to set up than the openstack configuration. First, you have to download a configuration directory, that is packaged as a zip file. This file can be found at

- https://openstack.tacc.chameleoncloud.org/dashboard/project/access_and_security/api_access/ec2/

Let us assume you have the username albert and the project FG-101. Than the zip file will be called:

```
FG-101-x509.zip
```

Let us set some environment variables to make the configuration description easier

```
$ export C_USERNAME=<your chameleon username>
$ export C_PROJECT=<your chameleon project name>
```

Unpack the zip file and place the entire directory in the .cloudmesh directory with. (We assume that you are in the directory where your browser downloaded the zip file and you have uncompressed it)

```
$ mkdir ~/.cloudmesh/clouds/chameleon/$C_PROJECT
$ cp $C_PROJECT ~/.cloudmesh/clouds/chameleon/$C_PROJECT
$ ls ~/.cloudmesh/clouds/chameleon/$C_PROJECT
```

The directory should include the files:

```
cacert.pem
cert.pem
ec2rc.sh
pk.pem
```

Take a look at the ec2rc.sh file

```
$ cat ~/.cloudmesh/clouds/chameleon/$C_PROJECT/ec2rc.sh
```

Now you can edit the cloudmesh yaml file at:

```
~/.cloudmesh/cloudmesh.yaml
```

locate the chameleon-ec2 entry and change the TBD values with the values you see in the ec2rc.sh file:

```
EC2_ACCESS_KEY: <find the value in the ec2rc.sh file>
EC2_SECRET_KEY: <find the value in the ec2rc.sh file>
EC2_USER_ID: <find the value in the ec2rc.sh file>
```

For the following lines in the cloudmesh file, please replace the TBD values with the chameleon project ID that you use for this cloud:

```
EC2_PRIVATE_KEY: ~/.cloudmesh/clouds/chameleon/TBD/pk.pem
EC2_CERT: ~/.cloudmesh/clouds/chameleon/TBD/cert.pem
NOVA_CERT: ~/.cloudmesh/clouds/chameleon/TBD/cacert.pem
EUCALYPTUS_CERT: ~/.cloudmesh/clouds/chameleon/TBD/cacert.pem
```

6.3.5 Registration of CloudLab Openstack Cloud

Todo

not yet tested but should work. add cloud registration here

6.3.6 Registration of AWS Cloud

Todo

not yet supported but used to be so we work on it ASAP. add cloud registration here

6.3.7 Registration of Azure Cloud

Todo

not yet supported but used to be so we work on it ASAP. add cloud registration here

6.3.8 Registration of devcloud

Todo

not tested, but should work as is regular openstack. add cloud registration here

6.3.9 Registration of a libcloud available cloud

Todo

not yet supported. add cloud registration here

CHAPTER 7

User Manual

In this section we summarize a number of commands that are useful for managing your multiple clouds. We organize them in the way you would use them in some order while:

- registering clouds
- creating virtual machines
- creating virtual clusters
- creating platforms on the clusters

If you have additional needs we provide a detailed list of man pages in alphabetical order in the Section ...

7.1 Shell Commands

7.1.1 Basic Commands and Options

Cloudmesh contains a number of commands that makes the management of multiple heterogeneous clouds easier. In order to better manage the various clouds it is convenient to introduce a number of options and behaviors. This includes the following concepts.

Format

Many commands have a format parameter that allows to provide output of the command in various formats. These formats include:

- json
- yaml
- table
- csv

The format can be changed on each command that supports it with:

```
--format FORMAT
```

where FORMAT is one of the values from the list above.

Todo

setting a default format via defaults

Not yet done: It is also possible to set the default format for all commands that accept the format option. THis is done with the command:

```
$ default format FORMAT
```

Once you have set it, the default format will be used for all commands the do not explicitly set the format option on the commandline.

To switch off this behavior and use the build in behavior for each command, we specify:

```
$ default format False
```

Cloud

Many commands are specific to a particular cloud. this cloud can be set with the:

```
--cloud CLOUD
```

option for individual commands that support it. As we deal with many clouds it may be inconvenient to specify the name of the cloud every time, thus we have introduced the concept of a default cloud. The default cloud can be set with the command:

```
$ default cloud CLOUDNAME
```

where cloudname is the name of the cloud that we have registered with cloudmesh (see registration).

Todo

put link to registration here

History

The manual page of the history command can be found at: register

Not yet completed. As we may want to run multiple commands we also provide a history that can be invoked from cloudmesh to show which cloudmesh commands have been issued in the past. This allows a more easy review of past activities:

```
$ cm history
```

Commands in history are preceeded by a number. A past command can be reissued by appending the number to the history. Thus the command:

```
$cm history 3
```

would execute the 3rd command in the command history. Instead of using the command history, you can also use the abbreviation *h*.

Help

To see the list of all available commands use the command:

```
$ cm help
```

The commands are sorted by topic, while the first list gives all commands in alphabetical order. To opbtain an individual man page simply say:

```
$ cm help COMMAND
```

where command is the command you which to get the help message for. To optain the manual pages of all commands yo can use the command:

```
$cm man
```

which will print all man pages out.

Shell & Commandline

Cloudmesh client is a shell as well as a commandline tool. Thus all commands that you can type in as a single command could also be executed as a command shell. To enter the command shell, please type:

```
$cm

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

cm>
```

You will see the prompt and can interactively execute some of the commands without needing to type in cm in front of each command. To see the commands type help. To get help for an individual command type help COMMANDNAME. You can quit the comamnd shell with the command quit.

The current list of commands contains:

```
cm help

Documented commands (type help <topic>):
========================================
EOF     cloud    group      key     man   pause  quota        secgroup  ssh
banner  context  help       limits  nova  q      register     select    version
clear   default  inventory  list    open  quit   reservation  server    vm

Security Commands
=================
key  secgroup  ssh

Shell Commands
==============
banner  clear  EOF  man  man  open  q

System Commands
===============
inventory  reservation
```

```
Cloud Commands
==============
cloud  default  group  limits  list  nova  quota  register  select  server  vm
```

Elementary Commands

We have build in some convenience commands into the shell that include comments and execution of cm scripts.

Comments

Comments are identified by the first characters in a command line. We allow the following comment charater identification strings:

```
#
/*
//
```

If comments are to be done over multiple lines in a cloudmesh script, they have to be done for each line. If a space or other character is in front of a comment string, the it will not be considered as a comment.

Cloudmesh File Execution

Multiple cloudmesh commands can be placed in a single file. We recommend that you use the ending *.cm*. You can satrt the execution of such a file with:

```
cm filename.cm
```

A cloudmesh file could itself include references to other cloudmesh files. They can be started in one of two ways. You can use the *exec* command

$ cm cm> exec filename.cm

or you can use simply the filename. Cloudmesh will check if the filename exists and than execute it:

```
$ cm
cm> filename.cm
```

Variables

CMD3 contains the ability to use variables within the shell. To see a list of all variables, use the command:

```
var list
```

or simply:

```
var
```

To use the content of the variable, simple use it on the shell with a dollar sign such as:

```
$date
```

Note that the variables $dat and $time are predefined and give the current date and time.

To set variable values you can use:

var name=value

Which will set the variable with the given name to the specified value. In case the value specifies an entry in the cloudmesh.yaml file it will be read from it and put into the named variable. For example the command

var username=cloudmesh.profile.username

Will create a variable username and get the value form the yaml file specified by its object hierarchy.

Note: Variables are not stored in the persistent database and have to be recreated every time a script is run.

Python

You can execute a python command as follows:

```
py COMMAND
```

where command is the command you like to execute

Quitting the shell

To quit the shell you can use either the commands:

```
q
quit
EOF
```

Manual Pages

Often you will run in the situation where you may have to create a list of manual pages for your commands for your users. To simplify that we have not provided this in Unix Man format, but simply in RST format. You can type in the command:

```
man
```

and it will print you in RST format a list of all commands available to you for your cmd3 shell. This naturally you could put into a sphinx documentation to create a nice user manual for your users.

Scripts

Cloudmesh can easily execute multiple cloudmesh commands that are stored in cloudmesh script files. TO do so we recommend to place them in a file ending with *.cm*. Let us assume we call the file test.cm.

Now we can simply execute the script with:

```
cm test.cm
```

you can also cat the file with

cat test.cm | cm

7.1.2 Color Command

You can toggle the color of the cloudmesh shell console by using the color command.

The manual page of the group command can be found at: group

Color ON/TRUE

Turn the color mode ON:

```
$ cm color ON
  Color True

$ cm color TRUE
  Color True
```

Color OFF/FALSE

Turn the color mode OFF:

```
$ cm color OFF
  Color False

$ cm color FALSE
  Color False
```

7.1.3 Default Command

The manual page of the group command can be found at: default

Cloudmesh has the ability to manage easily multiple clouds. One of the key concepts to make the list of such clouds easier is the introduction of defaults for each cloud or globally. Hence it is possible to set default images, flavors for each cloud, and also create the default cloud. The default command is used to set and list the default values. These defaults are used in other commands if they are not overwritten by a command parameter.

Upon start of cloudmesh, the default for cloud will be set to the first cloud that is found in the yaml file and the default group is set to *general*.

default list

All the current default values can by listed with –all option:

```
$ default list --all
+--------+-----------+-------+-------+
| user   | cloud     | name  | value |
+--------+-----------+-------+-------+
| albert | chameleon | image | abc   |
| albert | general   | cloud | azure |
| albert | general   | image | zyx   |
+--------+-----------+-------+-------+
```

You can also add a –cloud=CLOUD option to see the defaults set for a cloud:

```
$ default list --cloud=chameleon
+--------+-----------+-------+-------+
| user   | cloud     | name  | value |
+--------+-----------+-------+-------+
| albert | chameleon | image | abc   |
+--------+-----------+-------+-------+
```

set default values

To add a default value, type in a key=value pair. If no –cloud is specified, it adds the value to the general/global cloud:

```
$ default image=xyz
Successfully added value: xyz for key: image
```

With the –cloud=CLOUD option, defaults can be set for a particular cloud:

```
$ default image=xyz --cloud=chameleon
Successfully added value: xyz for key: image
```

looking up default values

To loop up a default value set, type in the key. If no –cloud option is specified, it returns the value of the general/global cloud:

```
$ cm default image
Default value for image is xyz
```

With the –cloud=CLOUD option, defaults can be looked up for a particular cloud:

```
$ default image --cloud=chameleon
Default value for image is xyz
```

deleting default values

To delete a default value, type in delete followed by the key. If no –cloud option is specified, it deletes the value of the general/global cloud:

```
$ default delete image
Deleted key image for cloud general
```

With the –cloud=CLOUD option, defaults can be deleted for a particular cloud:

```
$ default delete image --cloud=chameleon
Deleted key image for cloud chameleon
```

7.1.4 Group Command

One of cloudmesh major functionality is to group cloud and other resources into a named group. Such named groups can than be used to perform actions on them. Upon start the default group is set to general if no default group exists.

> **Warning:** at this time we have limited to groups to just hold ID of vms.

The manual page of the group command can be found at: group

Group List

The named groups can be listed with the following command:

```
$ cm group list --cloud india --format table
  +----------+-------+--------+----------+------+
  | user     | cloud | name   | value    | type |
  +----------+-------+--------+----------+------+
  | albert   | india | groupA | test-001 | vm   |
  | albert   | india | groupA | test-002 | vm   |
  | albert   | india | groupA | test-004 | vm   |
  | albert   | india | groupB | test-003 | vm   |
  | albert   | india | groupB | test-005 | vm   |
  +----------+-------+--------+----------+------+
```

Group Info

To get details about a particular group with specific name you can use the info option:

```
$ cm group list groupA
  +----------+-------+--------+----------+------+
  | user     | cloud | name   | value    | type |
  +----------+-------+--------+----------+------+
  | albert   | india | groupA | test-001 | vm   |
  | albert   | india | groupA | test-002 | vm   |
  | albert   | india | groupA | test-004 | vm   |
  +----------+-------+--------+----------+------+
```

Group Remove ID

To remove a VM from a particular group, you can use the remove option:

```
$ cm group remove --name groupA --id test-002
  Successfully removed ID [test-002] from the group [groupA]

$ cm group list groupA
  +----------+-------+--------+----------+------+
  | user     | cloud | name   | value    | type |
  +----------+-------+--------+----------+------+
  | albert   | india | groupA | test-001 | vm   |
  | albert   | india | groupA | test-004 | vm   |
  +----------+-------+--------+----------+------+
```

Group Add

To add a vm resource with specified id to a group with given name:

```
$ cm group add groupA --id test-001 --type vm
Created a new group [groupA] and added ID [test-001] to it

$ cm group info groupA
  +-----------+---------+--------+----------+------+
  | user      | cloud   | name   | value    | type |
  +-----------+---------+--------+----------+------+
  | albert    | general | groupA | test-001 | vm   |
  +-----------+---------+--------+----------+------+
```

Group Copy

To copy the VM(s) from one group to another use the command:

```
$ cm group copy groupA groupB
Created a new group [groupB] and added ID [test-001] to it

$ cm group info groupB
  +-----------+---------+--------+----------+------+
  | user      | cloud   | name   | value    | type |
  +-----------+---------+--------+----------+------+
  | albert    | general | groupB | test-001 | vm   |
  +-----------+---------+--------+----------+------+
```

Group Merge

Groups can be merged as follows:

```
$ cm group merge group01 groupB groupC
Merge of group [group01] & [groupB] to group [groupC] successful!

$ cm group info groupC
  +-----------+---------+--------+------------+------+
  | user      | cloud   | name   | value      | type |
  +-----------+---------+--------+------------+------+
  | albert    | general | groupC | albert-001 | vm   |
  | albert    | general | groupC | albert-002 | vm   |
  | albert    | general | groupC | test-001   | vm   |
  +-----------+---------+--------+------------+------+
```

Group Delete

A named group can be easily deleted.:

```
$ cm group delete groupC
Request to delete server albert-001 has been accepted.
Request to delete server albert-002 has been accepted.
Request to delete server test-001 has been accepted.
Deletion Successful!

$ cm group list groupC
ERROR: No group with name groupC found in the cloudmesh database!
```

Warning: When a group is deleted, all the instances (vms) are deleted, and a deletion request is submitted to the appropriate cloud.

7.1.5 Color Command

Often the ssh command needs to be used to login to remote machines. As the interaction with such machines could be frequent via the ssh command, it is often a good idea to include them into the ~/.ssh/config file. To simplify interaction, we provide a simple ssh command in cloudmesh.

The manual page of the group command can be found at: ssh

Lists

```
ssh list
    lists the hostsnames  that are present in the
    ~/.ssh/config file

ssh cat
    prints the ~/.ssh/config file

ssh table
    prints contents of the ~/.ssh/config file in table format
```

Executing Command

```
ssh ARGUMENTS
    executes the ssh command with the given arguments
    Example:
        ssh myhost

            conducts an ssh login to myhost if it is defined in
            ~/.ssh/config file
```

Register

```
ssh register NAME PARAMETERS
    registers a host i ~/.ssh/config file
    Parameters are attribute=value pairs
    Note: Note yet implemented
```

7.1.6 Select Command

Select Command is used to interactively set a default image/ flavor/ cloud/ key.

The manual page of the key command can be found at: SELECT

Setting default image

You can select the default image with the following simple command:

```
$ cm select image

Select an Image
===============

    1 - image-1
    2 - fedora
    3 - CentOS7
    4 - ubuntu-custom
    5 - Ubuntu-15.10-64
    6 - Ubuntu-14.04-64
    7 - cirros
    q - quit
```

```
Select between 1 - 7: 5
choice 5 selected.
Selected image Ubuntu-15.10-64
```

Setting default flavor

You can select the default flavor with the following simple command:

```
$ cm select flavor

Select a Flavor
===============

    1 - tiny
    2 - small
    3 - medium
    4 - large
    5 - xlarge
    q - quit


Select between 1 - 5: 3
choice 3 selected.
Selected flavor medium
```

Setting default cloud

You can select the default cloud with the following simple command:

```
$ cm select cloud

Select a cloud
==============

    1 - kilo
    2 - chameleon
    3 - cybera-c
    4 - cybera-e
    5 - aws
    6 - chameleon-ec2
    7 - azure
    q - quit



Select between 1 - 7: 2
choice 2 selected.
Selected cloud chameleon
```

Setting default key

You can select the default key with the following simple command:

```
$ cm select key

Select a Key
============

    1 - albert-key
    2 - customkey
    q - quit


Select between 1 - 2: 1
choice 1 selected.
Selected key albert-key
```

7.2 Cloud Commands

7.2.1 Register Command

Registering different clouds with the cloudmesh register command is easy. We have a number of predefined templates that are stored in the ~/.cloudmesh .yaml file that you can use and modify. However for some clouds such as the

once at IU an easy registration exists if you have appropriate access.

The manual page of the register command can be found at: register

Quickstart for registration of some clouds

Please only use the quickstart if you know hat you are doing, otherwise, read the manual. We assume you have access to the specific clouds that you like to access. On a terminal say:

```
cm register remote kilo
```

to register the FutureSystems kilo cloud

More information about the cloud can be found at

- https://portal.futuresystems.org

To register an openstack cloud for which you have an existing openrc.sh file, you can simply say:

```
cm register openrc.sh
```

Todo

verify if this works

On chameleoncloud.org you can for example go to the horizon web interface and download the credentials in teh security panel.

Introduction

As we are managing multiple clouds with cloudmesh we need to register them first. To make it easy for you cloudmesh reads the registered clouds from an easy to manage yaml file. This yam file is installed by default into the file:

```
~/.cloudmesh/cloudmesh.yaml
```

A number of templates in that file exist that refer to commonly used clouds. YOu can fill out the yaml file with your information, add new clouds, or delete templates of clouds that you do not use. We have several different types of clouds that we support. This includes OpenStack, AWS, and Azure clouds.

Todo

at this time we have not integrated our AWS and Azure IaaS abstractions in the new cloudmesh client. We will make them available in future.

As it may be inconvenient to edit this file and look at the yaml format, we provide several administrative commands. The command:

```
$ register info

File /Users/albert/.cloudmesh/cloudmesh.yaml exists. ok.
```

identifies if the *cloudmesh.yaml* file exists.

To view the contents of that file, you can cat it or use the command:

```
register cat
```

To edit the file, you can use the command:

```
register edit
```

register list

To list the clouds that are defined in the cloudmesh.yaml file, you can use the command:

```
$ register list
```

which will print a table with elementary information defined for the clouds.:

```
$ register list
Clouds specified in the configuration file ~/.cloudmesh\cloudmesh.yaml

+-------+-----------+---------+
| Name  | Iaas      | Version |
+-------+-----------+---------+
| azure | azure     | N/A     |
| aws   | ec2       | N/A     |
| kilo  | openstack | kilo    |
+-------+-----------+---------+
```

To list only the names, please use the command:

```
$ register list --name
Clouds specified in the configuration file ~/.cloudmesh\cloudmesh.yaml

+-------+
| Name  |
+-------+
| azure |
| aws   |
```

```
| india |
| kilo  |
+-------+
```

As we also have to sometimes login to some remote hosts it is convenient to reuse the ssh command for that. ssh has the advantage of being able to use a config file in $HOME/.ssh/config. MOre information about ssh config files and their format can be found in the many web pages if you google for *ssh config*. In case you have defined a host *india* in ~/.ssh/config in the following way:

Host india Hostname india.futuresystems.org User yourusername

The list command followed by ssh will give you a list of hosts defined in that file:

```
$ cm register list ssh

india
```

register remote

In case you already use an openstack cloud you may have come across an openrc.sh file. We are providing some very special helper functions, like for example obtain the openrc files from the FutureSystems cloud.

The command:

```
register remote HOSTNAME
```

will copy and register a machine on which an openrc.sh file is located into the *cloudmesh.yaml* file. With cloudmesh we provide some default host, thus

they are very easy to configure. This includes *kilo* our current clouds in our lab. To register them you can use the commands:

```
    cm register reomte kilo
```

These commands will only work if you have an account on this machine and it is integrated into the ssh config file as discussed previously.

register export

To view the data associated with a particular cloud you can just use the command export:

```
register export kilo --format=table
```

Which will look like this:

```
+----------------------+------------------------------------------+
| Attribute            | Value                                    |
+----------------------+------------------------------------------+
| OS_PASSWORD          | ********                                 |
| OS_VOLUME_API_VERSION | 2                                       |
| OS_IMAGE_API_VERSION | 2                                        |
| OS_PROJECT_DOMAIN_ID | default                                  |
| OS_USER_DOMAIN_ID    | default                                  |
| OS_TENANT_NAME       | fg1234                                   |
| OS_PROJECT_NAME      | fg1234                                   |
| OS_USERNAME          | albert                                   |
| OS_AUTH_URL          | https://kilo.futuresystems.org:5000/v3   |
| OS_VERSION           | kilo                                     |
```

```
| OS_OPENRC             | ~/.cloudmesh/clouds/india/kilo/openrc.sh |
+-----------------------+------------------------------------------+
```

The default view returns a openrc.sh file:

```
cm register export kilo
```

The output contains an rc file example:

```
export OS_PROJECT_DOMAIN_ID=default
export OS_USERNAME=albert
export OS_OPENRC=~/.cloudmesh/clouds/india/kilo/openrc.sh
export OS_AUTH_URL=https://kilo.futuresystems.org:5000/v3
export OS_TENANT_NAME=1234
export OS_USER_DOMAIN_ID=default
export OS_VERSION=kilo
export OS_VOLUME_API_VERSION=2
export OS_IMAGE_API_VERSION=2
export OS_PASSWORD=********
export OS_PROJECT_NAME=fg1234
```

The passwords will be masked with eight stars: ********. In case you like also to see the password you can use the –password flag.

register merge

Todo

the description of what this is doing was ambigous, we need to clarify if it only replaces to do or actually add things that do not exist, or just overwrites.

IN case you have already a yaml file, form another project you can merge two of them into the same cloudmesh yaml file. You simply have to specify the location of the file that you like to merge into the existing yaml file. However, please be careful, as it will overwrite the contents in ~/.cloudmesh/cloudmesh.yaml

---

Todo

We used to have a .bak.# when we modified the yaml file, do you still have this

---

Hence the command

$ cm register merge my_cloudmesh.yaml

This command allows the content from another yaml file to be merged into the regular cloudmesh.yaml file. A backup of the old cloudmesh.yaml file is created with an increased number.

register form

In some cases it is nice to have an interactive mechanism to fill out the missing yaml file information that is indicated with TBD. THis is useful, if you do not have an editor at hand. Thus you can use the command:

```
register form
```

It will interactively fills out the form wherever we find TBD:

```
$ cm register form
Please enter email[TBD]:
Editing the credentials for cloud india
Please enter OS_TENANT_NAME[TBD]:
Editing the credentials for cloud aws
Please enter EC2_ACCESS_KEY[TBD]:
Please enter EC2_SECRET_KEY[TBD]:
Please enter keyname[TBD]:
Please enter userid[TBD]:
Editing the credentials for cloud azure
Please enter managementcertfile[TBD]:
Please enter servicecertfile[TBD]:
Please enter subscriptionid[TBD]:
Please enter thumbprint[TBD]:
```

register check

o find any not filled out values, you can use the command:

```
register check
```

which hecks the yaml file for completness and list all fields that have the value TBD:

```
$ cm register check
ERROR: The file has 11 values to be fixed

  email: TBD
  username: TBD
  flavor: TBD
  EC2_ACCESS_KEY: TBD
  EC2_SECRET_KEY: TBD
  keyname: TBD
  userid: TBD
  managementcertfile: TBD
  servicecertfile: TBD
  subscriptionid: TBD
  thumbprint: TBD
```

register json HOST

Instead of using the cat command and listing the contents of a cloud registration in yaml format you can also explicitly obtain a jason representation by issueing the command:

```
register json
```

It will return output in json format:

```
$ cm register json azure
{
    "cm_heading": "Microsoft Azure Virtual Machines",
    "cm_label": "waz",
    "cm_host": "windowsazure.com",
    "default": {
        "flavor": "ExtraSmall",
        "image": "b39f27a8b8c64d52b05eac6a62ebad85__Ubuntu-14_04_2-LTS-amd64-server-20150610-en-us-30
        "location": "East US"
```

```
    },
    "credentials": {
        "managementcertfile": "TBD",
        "servicecertfile": "TBD",
        "subscriptionid": "TBD",
        "thumbprint": "TBD"
    },
    "cm_type": "azure",
    "cm_type_version": null
}
```

7.2.2 Cloud Command

The cloud command provides an API that allows users to login to a cloud, activate a cloud, deactivate a cloud & logout from a cloud.

The manual page of the network command can be found at: *cloud <../man/man.html#cloud>*

List status of all clouds

To list status of all clouds registered in the cloudmesh.yaml file use:

```
cm> cloud list
  +------------+------------+
  | cloud name | status     |
  +------------+------------+
  | aws        | Logged Out |
  | azure      | Logged Out |
  | chameleon  | Logged Out |
  | kilo       | Logged Out |
  +------------+------------+
```

Login to a single/multiple clouds

To logon to a cloud use:

```
cm> cloud logon kilo
  Logged into cloud: kilo
```

You can logon to multiple clouds:

```
cm> cloud logon kilo
  Logged into cloud: kilo

cm> cloud list
  +------------+------------+
  | cloud name | status     |
  +------------+------------+
  | aws        | Logged Out |
  | azure      | Logged Out |
  | chameleon  | Logged Out |
  | kilo       | Active     |
  +------------+------------+
```

Deactivate a cloud

To deactivate a cloud use:

```
cm> cloud deactivate kilo
  Deactivated cloud: kilo

cm> cloud list
  +------------+------------+
  | cloud name | status     |
  +------------+------------+
  | aws        | Logged Out |
  | azure      | Logged Out |
  | chameleon  | Logged Out |
  | kilo       | Inactive   |
  +------------+------------+
```

Activate a cloud

To activate a cloud use:

```
cm> cloud activate kilo
  Activated cloud: kilo

cm> cloud list
  +------------+------------+
  | cloud name | status     |
  +------------+------------+
  | aws        | Logged Out |
  | azure      | Logged Out |
  | chameleon  | Logged Out |
  | kilo       | Active     |
  +------------+------------+
```

Log out from a cloud

To log out from a cloud use:

```
cm> cloud logout kilo
  Logged out of cloud: kilo

cm> cloud logout kilo
  Logged out of cloud: kilo

cm> cloud list
  +------------+------------+
  | cloud name | status     |
  +------------+------------+
  | aws        | Logged Out |
  | azure      | Logged Out |
  | chameleon  | Logged Out |
  | kilo       | Logged Out |
  +------------+------------+
```

7.2.3 Key Command

In clouds and distributed environments security keys are used for authentication. We like to be able to register specific keys with clouds or vms and easily use them. To do so we upload them into a key registry in which each key is uniquely named. We use these named keys when we start up virtual machines or log into remote machines.

The manual page of the key command can be found at: key

Adding a key to the database

To add a key to the key registry from a file we use the command:

```
$ cm key add --name=demokey /home/albert/key_expt/id_rsa.pub
Key demokey successfully added to the database
info. OK.
```

List Keys

To list the keys in the registry you can use the command:

```
$ cm key list
  +---------+--------------------+-------------------------------------------+--------------------
  | name    | comment            | uri                                       | fingerprint
  +---------+--------------------+-------------------------------------------+--------------------
  | demokey | albert@Zweistein | file:///home/key_expt/id_rsa.pub            | 4e:fc:e8:03:4e:c7:8e
  +---------+--------------------+-------------------------------------------+--------------------
  info. OK.
```

The key command takes a number of additional options. Instead of using the cloudmesh registry, keys can also be read from git hub with the option:

```
$ cm key list --source=git
+------+----------+----------------------------+-------------------------------------------------+--
| name | comment  | uri                        | fingerprint                                     | s
+------+----------+----------------------------+-------------------------------------------------+--
|      | github-0 | https://github.com/TBD.keys | 6e:95:48:8d:af:20:75:2a:52:6b:c5:29:d3:71:0a:8b |
|      | github-1 | https://github.com/TBD.keys | 8a:4f:fe:80:be:e5:ec:c8:c1:1d:e9:74:28:41:c5:a3 |
+------+----------+----------------------------+-------------------------------------------------+--
info. OK.
```

To change the output format you can specify it with the –format option:

```
$ cm key list --source=git --format=json
{
    "github-0": {
        "comment": "github-0",
        "string": "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC/4dvq0KG++Tieu4vhqL4WptgsSUIq+vqLi4PiR6N+UE
        "uri": "https://github.com/TBD.keys",
        "key": "AAAAB3NzaC1yc2EAAAADAQABAAABAQC/4dvq0KG++Tieu4vhqL4WptgsSUIq+vqLi4PiR6N+UBwEcYWzX33O
        "fingerprint": "6e:95:48:8d:af:20:75:2a:52:6b:c5:29:d3:71:0a:8b",
        "type": "ssh-rsa",
        "Id": "github-0"
    },
    "github-1": {
        "comment": "github-1",
        "string": "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDNTRjYstjHaZyS+vOssLOxYv57z1YEndk5VI34PFb6zk
        "uri": "https://github.com/TBD.keys",
```

```
        "key": "AAAAB3NzaC1yc2EAAAADAQABAAABAQDNTRjYstjHaZyS+vOssLOxYv57z1YEndk5VI34PFb6zb9JI3kTZ0wvh
        "fingerprint": "8a:4f:fe:80:be:e5:ec:c8:c1:1d:e9:74:28:41:c5:a3",
        "type": "ssh-rsa",
        "Id": "github-1"
    }
}
info. OK.
```

Get Keys

To get the fingerprint of a key you can obtain it with:

```
$ cm key get demokey
   demokey: 4e:fc:e8:03:4e:c7:8e:ca:30:1a:54:43:8d:24:90:39
   info. OK.
```

Default Keys

In many cases it is convenient to just use a default key that is set. To mark key as default by name you can use the command:

```
$ cm key default demokey
Key demokey set as default
info. OK.
```

You can verify that a key is set as default while looking at the 'is_default' attribute:

```
$ cm key list --format=json

       "1": {
           "comment": "albert@Zweistein",
           "is_default": "True",  <<--Set to True
           "kind": "key",
           "name": "demokey",
           "created_at": "2015-09-23 15:58:32",
           "uri": "file:///home/key_expt/id_rsa.pub",
           "value": null,
           "updated_at": "2015-09-23 16:14:41",
           "project": "undefined",
           "source": "ssh",
           "user": "undefined",
           "fingerprint": "4e:fc:e8:03:4e:c7:8e:ca:30:1a:54:43:8d:24:90:39",
           "label": "demokey",
           "id": 1,
           "cloud": "general"
       }
   }
   info. OK.
```

To make it easy for the user, we can set the default key also interactively with the select option:

```
$ cm key default --select

KEYS
====
```

```
    1 - demokey: 4e:fc:e8:03:4e:c7:8e:ca:30:1a:54:43:8d:24:90:39
    2 - rsa: 2d:18:a8:03:1e:e1:7e:fe:b3:fa:59:49:c7:c2:cf:01
    q - quit


Select between 1 - 2: 2
choice 2 selected.
Setting key: rsa as default.
info. OK.
```

Delete Keys

A named key can be deleted from the registry with the command, where 'demokey' is the name of the key:

```
$ cm key delete demokey
Key demokey deleted successfully from database.
info. OK.
```

Alternatively you can also interactively select it:

```
$ cm key delete --select

KEYS
====

    1 - rsa: 2d:18:a8:03:1e:e1:7e:fe:b3:fa:59:49:c7:c2:cf:01
    2 - demokey: 4e:fc:e8:03:4e:c7:8e:ca:30:1a:54:43:8d:24:90:39
    q - quit


Select between 1 - 2: 2
choice 2 selected.
Deleting key: demokey...
info. OK.
```

To delete all keys from database use:

```
$ cm key delete --all
All keys from the database deleted successfully.
info. OK.
```

Adding Key to Cloud

This functionality is required for key management with VMs. We can add the key from database to the target cloud.:

```
$ cm key add_to_cloud albertkey
Adding key albertkey to cloud kilo as albert-kilo-albertkey
Key albertkey added successfully to cloud kilo as albert-kilo-albertkey.
info. OK.
```

By default the target cloud key name format is <username>-<cloud>-<key-name>. However, you may choose to override it with '–name_on_cloud' argument.:

```
$ cm key add_to_cloud albertkey --name_on_cloud=someothername
key add_to_cloud albertkey --name_on_cloud=someothername
Adding key albertkey to cloud kilo as someothername
```

```
Key albertkey added successfully to cloud kilo as someothername.
info. OK.
```

List Key Cloud Mapings

You may check out the mappings of database key names with the cloud key names.:

```
$ cm key list_cloud_mappings
+-----------+-----------+------------+-------------------------+
| user      | key_name  | cloud_name | key_name_on_cloud       |
+-----------+-----------+------------+-------------------------+
| albert    | albertkey | kilo       | albert-kilo-albertkey   |
+-----------+-----------+------------+-------------------------+
```

7.2.4 List Command

The cloudmesh list command provides you with the ability to easily list information in regards to virtual machines, images, flavors, defaults, and available clouds.

The manual page of the list command can be found at: list

List Default

To list all default values you can use:

```
$ cm list --cloud general default
```

To list the default values set in a particular cloud use:

```
$ cm list --cloud general default
  +-----------+---------+--------+----------+----------------------------+----------------------------
  | user      | cloud   | name   | value    | created_at                 | updated_at
  +-----------+---------+--------+----------+----------------------------+----------------------------
  | albert    | general | tenant | fg478    | 2015-09-21 02:24:31.978000 | 2015-09-21 02:24:31.978000
  | albert    | general | cloud  | india    | 2015-09-21 02:25:00.781000 | 2015-09-21 02:25:00.781000
  | albert    | general | group  | group001 | 2015-09-23 21:53:04        | 2015-09-23 21:53:04
  | albert    | general | format | table    | 2015-09-23 21:53:16        | 2015-09-23 21:53:16
  +-----------+---------+--------+----------+----------------------------+----------------------------
```

To specify a different format, such as json, use:

```
$ cm list --cloud general --format json default
{
      "1": {
          "cloud": "general",
          "created_at": "2015-09-21 02:24:31.978000",
          "id": "1",
          "kind": "default",
          "label": "tenant",
          "name": "tenant",
          "project": "undefined",
          "type": "string",
          "updated_at": "2015-09-21 02:24:31.978000",
          "user": "albert",
          "value": "fg478"
```

```
    },
    "2": {
        "cloud": "general",
        "created_at": "2015-09-21 02:25:00.781000",
        "id": "2",
        "kind": "default",
        "label": "cloud",
        "name": "cloud",
        "project": "undefined",
        "type": "string",
        "updated_at": "2015-09-21 02:25:00.781000",
        "user": "albert",
        "value": "india"
    },
    "3": {
        "cloud": "general",
        "created_at": "2015-09-23 21:53:04",
        "id": "3",
        "kind": "default",
        "label": "group",
        "name": "group",
        "project": "undefined",
        "type": "string",
        "updated_at": "2015-09-23 21:53:04",
        "user": "albert",
        "value": "group001"
    },
    "4": {
        "cloud": "general",
        "created_at": "2015-09-23 21:53:16",
        "id": "4",
        "kind": "default",
        "label": "format",
        "name": "format",
        "project": "undefined",
        "type": "string",
        "updated_at": "2015-09-23 21:53:16",
        "user": "albert",
        "value": "table"
    }
}
```


list Cloud objects

The list command can also be used to list cloud objects, thus you can use:

```
list image
list flavor
list quota
list limits
list usage
list vm
```

7.2.5 SecGroup Command

A security group is a named collection of network access rules that are use to limit the types of traffic that have access to instances. When you launch an instance, you can assign one or more security groups to it. If you do not create security groups, new instances are automatically assigned to the default security group, unless you explicitly specify a different security group.

The associated rules in each security group control the traffic to instances in the group. Any incoming traffic that is not matched by a rule is denied access by default. You can add rules to or remove rules from a security group, and you can modify rules for the default and any other security group.

The manual page of the secgroup command can be found at: secgroup

Security Group Create

To create a security group in cloudmesh for a cloud and tenant use:

```
$ cm secgroup create --cloud india --tenant fg478 test-group02
Created a new security group [test-group02] with UUID [bd9cb15e-5fcf-11e5-85fd-d8eb97bdb464]
```

Security Group List

To list Security Groups in cloudmesh for a cloud and tenant use:

```
$ cm secgroup list --cloud india --tenant fg478
  +--------------------------------------+--------------------------------------+-------------
  | Id                                   | Name                                 | Description
  +--------------------------------------+--------------------------------------+-------------
  | 7ee21121-5fcc-11e5-8497-d8eb97bdb464 | albert-security_group-q5ukqwab4odq   | SSL(443), Web(5000)
  | 4bc8bbb1-014d-4a84-a62c-f216d620c2bc | albert-security_group-r2qpv3kefysi   | SSL(443), Web(5000)
  | 68c31654-7f5f-4944-a295-b9ff29a7e170 | albert-security_group-ayzancofltyf   | SSL(443), Web(5000)
  +--------------------------------------+--------------------------------------+-------------
```

Security Group Rule Add

To add a new rule to the security group use:

```
$ cm secgroup rules-add --cloud india --tenant fg478 test-group 80 80 tcp  0.0.0.0/0
  Added rule [80 | 80 | tcp | 0.0.0.0/0] to secgroup [test-group]

$ cm secgroup rules-add --cloud india --tenant fg478 test-group 443 443 udp  0.0.0.0/0
  Added rule [443 | 443 | udp | 0.0.0.0/0] to secgroup [test-group]
```

Security Group Rules List

To list all the rules assigned to the security group use:

```
$ cm secgroup rules-list --cloud india --tenant fg478 test-group
  +----------+-------+------------+----------+--------+----------+-----------+
  | user     | cloud | name       | fromPort | toPort | protocol | cidr      |
  +----------+-------+------------+----------+--------+----------+-----------+
  | albert   | india | test-group | 80       | 80     | tcp      | 0.0.0.0/0 |
  | albert   | india | test-group | 443      | 443    | udp      | 0.0.0.0/0 |
  +----------+-------+------------+----------+--------+----------+-----------+
```

Security Group Rule Delete

To delete a specific rule within a security group use:

```
$ cm secgroup rules-delete --cloud india --tenant fg478 test-group 80 80 tcp 0.0.0.0/0
  Rule [80 | 80 | tcp | 0.0.0.0/0] deleted

$ cm secgroup rules-list india fg478 test-group
  +----------+-------+--------------+----------+--------+----------+------------+
  | user     | cloud | name         | fromPort | toPort | protocol | cidr       |
  +----------+-------+--------------+----------+--------+----------+------------+
  | albert   | india | test-group   | 443      | 443    | udp      | 0.0.0.0/0 |
  +----------+-------+--------------+----------+--------+----------+------------+
```

Security Group Delete

To delete an entire security group use:

```
$ cm secgroup delete --cloud india --tenant fg478 test-group
  Rule [443 | 443 | udp | 0.0.0.0/0] deleted
  Security Group [test-group] for cloud [india], & tenant [fg478] deleted

$ cm secgroup rules-list --cloud india --tenant fg478 test-group
  ERROR: Security Group with label [test-group], cloud [india], & tenant [fg478] not found!
```

7.2.6 VM Command

VM Command is used to manage VM instances across clouds. It is like a one stop interface that can be used to perform various VM operations on various clouds available to Cloudmesh.

The manual page of the key command can be found at: VM

Listing Defaults

You can have a list of relevant default attributes required for VM operations:

```
+-----------+--------------------------------------+
| Attribute | Value                                |
+-----------+--------------------------------------+
| secgroup  |                                      |
| login_key | /home/albert/key/id_rsa              |
| flavor    | 2                                    |
| image     | 619b8942-2355-4aa2-jaa5-74b8f1751911 |
| cloud     | kilo                                 |
| name      | albert-015                           |
| key       | albertkey                            |
| group     | test                                 |
+-----------+--------------------------------------+
```

- secgroup - Security Group to be provided for VM boot.
- login_key - Path to private key required for VM login.
- flavor - Flavor ID required for VM boot.
- image - Image ID required for VM boot.

- cloud - Target Cloud.
- name - Name of the VM to be booted. This is in format <username>-<count>. Username retrieved from cloudmesh.yaml, count retrieved from a counter in database.
- key - Key name from db used for VM boot.
- group - Group for the VM to be booted.

Booting a VM instance

If you have all the required attributes (secgroup not mandatory) setup and listed in the vm defaults, then you can simply run the following to boot a vm.:

```
$ cm vm boot
Machine albert-015 is being booted on kilo Cloud...
Added ID [4a37b49a-9768-88cc-b988-01013701a8fb] to Group [test]
info. OK.
```

Else you may explicitly specify the attribute values in the arguments to the vm boot command.:

```
$ cm vm boot --name=testvm --cloud=kilo --image=619b8942-2355-4aa2-jaa5-74b8f1751911 --flavor=2
Machine testvm is being booted on kilo Cloud...
```

Listing a VM instances

You can list all the VM instances running on the cloud by 'vm list' command like the one below:

```
+----+--------------------------------------+------------------------------+------------+-------------
| id | uuid                                 | label                        | status     | static_ip
+----+--------------------------------------+------------------------------+------------+-------------
| 10 | 21305503-2649-3664-8876-d825758c83f3 | albert-001                   | ACTIVE     | 10.20.99.xx
| 9  | 94f01af3-ee2a-9887-b228-75627f358169 | albert-001                   | SHUTOFF    | 10.20.99.xx
| 8  | 2f275d38-62af-1223-a04a-0456e0d6466f | albert-server-jzqc23pekfcu   | SUSPENDED  | 10.20.99.xx
| 7  | 6730c273-609f-9879-a481-313ff4200d82 | albert-server-ekbvvsmjyqlo   | ACTIVE     | 10.20.99.xx
| 6  | fa3580f3-2dbd-d666-9178-326b39916c09 | albert-server-cdmelfaefggf   | ACTIVE     | 10.20.99.xx
+----+--------------------------------------+------------------------------+------------+-------------
```

Stop a VM

You can stop a VM by supplying it's label or UUID:

```
$ cm vm stop testvm --cloud=kilo
Machine testvm is being stopped on kilo Cloud...
info. OK.
```

Start a VM

You can start a VM by supplying it's label or UUID:

```
$ cm vm start testvm --cloud=kilo
Machine testvm is being started on kilo Cloud...
info. OK.
```

Assign Floating IP to VM

In order to access the vm from outside of the cloud private network, we need to assign a floating IP which can be accessed publicly:

```
$ cm vm floating_ip_assign testvm --cloud=kilo
Floating IP assigned to testvm successfully and it is: 149.165.158.XX
```

Retrieving IP Address details

You can get the IP address details of a VM by the following command:

```
$ cm vm ip_show testvm --cloud=kilo
IP Addresses of instance testvm are as follows:-
+---------+---------+----------------+
| network | version | addr           |
+---------+---------+----------------+
| int-net | 4       | 10.23.2.XX     |
| int-net | 4       | 149.165.158.XX |
+---------+---------+----------------+
```

Login to VM

You can login to a VM in your target cloud:

```
$ cm vm login testvm --user=albert --key=/location/id_rsa --cloud=kilo
Logging in into testvm machine...
Determining IP Address to use with a ping test...
Checking 10.23.2.XX...
Cannot reach 10.23.2.XX.
Checking 149.165.158.XX...
IP to be used is: 149.165.158.XX
Warning: Permanently added '149.165.158.XX' (ECDSA) to the list of known hosts.
Enter passphrase for key '/location/id_rsa':
Welcome to <OS> <VERSION>.3 LTS (GNU/Linux <VERSION> <BIT_SPEC>)

  * Documentation:  https://help.os.com/

  System information as of Mon Oct 19 04:17:48 UTC 2015

  System load: 0.0               Memory list: 2%   Processes:       52
  Usage of /:  56.9% of 1.32GB   Swap list:   0%   Users logged in: 0

  Graph this data and manage this system at:
    https://landscape.canonical.com/

  Get cloud support with OS Advantage Cloud Guest:
    http://www.OS.com/business/services/cloud

0 packages can be updated.
0 updates are security updates.



The programs included with the OS system are free software;
```

```
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

OS comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

albert@testvm:~$
```

Running command on VM

You can use the vm login to simply run a command on the target VM:

```
$ cm vm login testvm --user=albert --key=/location/id_rsa --command="uname\ -a" --cloud=kilo
Logging in into testvm machine...
Determining IP Address to use with a ping test...
Checking 10.23.2.XX...
Cannot reach 10.23.2.XX.
Checking 149.165.159.XX...
IP to be used is: 149.165.159.XX
Enter passphrase for key '/location/id_rsa':
OS testvm <VERSION> #103-OS SMP Fri Aug 14 21:42:59 UTC 2015 <BIT_SPEC> OS
```

Deleting a VM

You can delete a VM on the target cloud by using 'vm delete' command as below:

```
$ cm vm delete testvm --cloud=kilo
Machine testvm is being deleted on kilo Cloud...
```

7.2.7 Nova Command

This is a wrapper nova command provided by cloudmesh which in turn calls the openstack nova command on the target cloud. This also provides you with the capability of setting the target cloud. However, we recommend not using the command and instead use the cloudmesh command sas they allow for information caching

The manual page of the key command can be found at: Nova

Setting the Target Cloud

You may set the target cloud on which the nova command should run as follows:

```
$ cm nova set india
india is set
```

Note that if you do not set a target cloud, default cloud considered is 'india'.

Getting the Cloud Info

You may get the cloud info in the following manner:

```
$ cm nova info
WARNING: OS environment variable OS_REGION not found
+----------------+------------------------------------------------------------+
| Variable       | Value                                                      |
+----------------+------------------------------------------------------------+
| OS_REGION      | None                                                       |
| OS_USERNAME    | albert                                                     |
| OS_CACERT      | /home/albert/.cloudmesh/clouds/india/kilo/cacert.pem       |
| OS_TENANT_NAME | fg478                                                      |
| OS_AUTH_URL    | https://i5r.idp.iu.futuregrid.org:5000/v2.0                |
| OS_PASSWORD    | ********                                                   |
+----------------+------------------------------------------------------------+
```

By default it gives the 'india' cloud info. To check for specific cloud, here is an example for kilo cloud:

```
$ cm nova info kilo
+----------------+-----------------------------------------------------+
| Variable       | Value                                               |
+----------------+-----------------------------------------------------+
| OS_REGION      | None                                                |
| OS_USERNAME    | TBD                                                 |
| OS_CACERT      | TBD                                                 |
| OS_TENANT_NAME | TBD                                                 |
| OS_AUTH_URL    | https://i5r.idp.iu.futuresystems.org:5000/v2.0      |
| OS_PASSWORD    | ********                                            |
+----------------+-----------------------------------------------------+
```

Running openstack nova commands

The syntax is the same as what is used for openstack nova. Following are couple of examples:

Listing images:

```
$ cm nova image-list
Cloud = india
+--------------------------------------+-----------------------------------------+--------+----------
| ID                                   | Name                                    | Status | Server
+--------------------------------------+-----------------------------------------+--------+----------
| 619b8942-2355-4aa2-bae3-74b8f1751911 | CentOS-7                                | ACTIVE |
| f63a996c-ea69-4a56-830e-c190bca2f828 | VM with Cloudmesh Configured Completely | ACTIVE | 8b7ce3bf-
| 7ddc3366-73bf-453a-a813-43514030bf2e | badi/centos-7-2015-06-01                | ACTIVE |
| c3c5b676-be53-4237-a40f-451d4c6e572e | badi/ubuntu-14.04-2015-06-01            | ACTIVE |
| f2c2bbda-8bc1-4f02-a2e8-60014da66689 | cloudmesh/ipynb-n-java                  | ACTIVE |
| 186592ce-eed5-4631-bc0c-7022eccd8508 | fg464/hadoop-b649                       | ACTIVE | 63a2cf03-
| 364bd53b-87d3-4ac6-8e41-af540301f0cd | futuresystems/centos-7                  | ACTIVE |
| 58e5d678-79ec-4a4d-9aa8-37975b7f40ac | futuresystems/fedora-21                 | ACTIVE |
| a59833a2-60c9-47f0-b333-4e0bc071ac3a | futuresystems/hadoop-v2                 | ACTIVE | f01633b1-
| 367de5c7-3a30-4bad-b316-1a2afa17d794 | futuresystems/ubuntu-12.04              | ACTIVE |
| 66708636-5ed6-4908-b36a-f5a69f8ac7ee | futuresystems/ubuntu-14.04              | ACTIVE |
| 0f787e59-6ff9-466c-aaf6-cd3f3c9350d0 | kilitbilgi/ubuntu_14_10_desktop         | ACTIVE |
| 5337a50d-4418-4c1f-9741-5c31bf03e267 | lee212/CoreOS                           | ACTIVE |
| 132c961f-bca8-4942-a2c5-a8f60f84aea9 | lee212/CoreOS-Alpha                     | ACTIVE |
| e8acb8e0-fbc9-44e4-9b31-3c38fc9c25ae | lee212/boot2docker                      | ACTIVE |
| b073ddce-747d-4c66-8152-70118a4e5781 | mooc-backup                             | ACTIVE | 805da4cb-
| 85fdb68e-8bd3-4e5e-bb4e-f286298f4fe6 | said/ubuntu15                           | ACTIVE |
| e3d5fcf5-1b40-48df-9098-3c03a682421e | slaves_ubuntu_14_04                     | ACTIVE |
| 58c9552c-8d93-42c0-9dea-5f48d90a3188 | ubuntu12-cometworker1                   | ACTIVE | 55458942-
+--------------------------------------+-----------------------------------------+--------+----------
```

Listing flavors:

```
$ cm nova flavor-list
Cloud = india
+----+----------------+-----------+------+-----------+------+-------+-------------+-----------+
| ID | Name           | Memory_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor | Is_Public |
+----+----------------+-----------+------+-----------+------+-------+-------------+-----------+
| 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      |
| 6  | m1.small_e30   | 2048      | 20   | 30        |      | 1     | 1.0         | True      |
| 7  | m1.medium_e60  | 4096      | 40   | 60        |      | 2     | 1.0         | True      |
| 8  | m1.large_e100  | 8192      | 80   | 100       |      | 4     | 1.0         | True      |
| 9  | m1.xlarge_e200 | 16384     | 160  | 200       |      | 8     | 1.0         | True      |
+----+----------------+-----------+------+-----------+------+-------+-------------+-----------+
```

Following is the link for openstack nova command manual:-

Openstack nova command manual

7.2.8 Flavor Command

The manual page of the flavor command can be found at: Flavor

Flavors define the compute, memory, and storage capacity of nova computing instances. To put it simply, a flavor is an available hardware configuration for a server. It defines the size of a virtual server that can be launched

Refresh

The refresh command would update the local database with the latest flavors. To refresh flavors of a cloud, do the following:

```
$cm flavor refresh --cloud=kilo
Refresh flavor for cloud kilo. ok
```

List

To list the set of flavors of a cloud, do the following:

```
$cm flavor list --cloud=kilo
+----+----------------+--------+-------+----------+-------+------+--------+-------------+------------
| Id | Name           | User   | RAM   | Disabled | vCPUs | Swap | Access | rxtx_factor | os_flv_ext_
+----+----------------+--------+-------+----------+-------+------+--------+-------------+------------
| 1  | m1.tiny        | albert | 512   | 0        | 1     |      | 1      | 1.0         | 0
| 5  | m1.xlarge      | albert | 16384 | 0        | 8     |      | 1      | 1.0         | 0
| 9  | m1.xlarge_e200 | albert | 16384 | 0        | 8     |      | 1      | 1.0         | 200
| 2  | m1.small       | albert | 2048  | 0        | 1     |      | 1      | 1.0         | 0
| 6  | m1.small_e30   | albert | 2048  | 0        | 1     |      | 1      | 1.0         | 30
| 3  | m1.medium      | albert | 4096  | 0        | 2     |      | 1      | 1.0         | 0
| 7  | m1.medium_e60  | albert | 4096  | 0        | 2     |      | 1      | 1.0         | 60
| 4  | m1.large       | albert | 8192  | 0        | 4     |      | 1      | 1.0         | 0
| 8  | m1.large_e100  | albert | 8192  | 0        | 4     |      | 1      | 1.0         | 100
+----+----------------+--------+-------+----------+-------+------+--------+-------------+------------
```

List Details

To list the details of a flavor, give in the id, uuid or name of the flavor. In case latest information is needed, the –refresh option can be used which would update the local database:

```
$cm flavor list 1 --cloud=kilo
+-----------------+----------------------+
| Attribute       | Value                |
+-----------------+----------------------+
| id              | 1                    |
| swap            |                      |
| os_flv_disabled | 0                    |
| os_flv_ext_data | 0                    |
| disk            | 0                    |
| os_flavor_acces | 1                    |
| vcpus           | 1                    |
| uuid            | 1                    |
| rxtx_factor     | 1.0                  |
| created_at      | 2015-11-11 13:38:31  |
| updated_at      | 2015-11-11 13:38:31  |
| ram             | 512                  |
| user            | albert               |
| kind            | flavor               |
| cloud           | kilo                 |
| name            | m1.tiny              |
| label           | m1.tiny              |
| project         | undefined            |
+-----------------+----------------------+
```

7.2.9 Image Command

The manual page of the image command can be found at: Image

An image is a collection of files used to create or rebuild a server

Refresh

The refresh command would update the local database with the latest images. To refresh images of a cloud (in this example, kilo), do the following:

```
$cm image refresh --cloud=kilo
Refresh image for cloud kilo. ok.
```

List

To list the set of images of a cloud, do the following:

```
$cm image list --cloud=kilo
+----+--------------+----------------------+------------------------------+---------+--------+------
| id | size         | created              | description                  | minDisk | minRam | name
+----+--------------+----------------------+------------------------------+---------+--------+------
| 1  | 158443520    | 2015-03-23T20:50:29Z |                              | 0       | 0      | XXX
| 2  | 1270546432   | 2015-03-26T18:15:47Z |                              | 20      | 0      | YYY
| 3  | 4845404160   | 2015-03-26T20:05:29Z |                              | 40      | 0      | mooc-
+----+--------------+----------------------+------------------------------+---------+--------+------
```

List Details

To list the details of an image, give in the id, uuid or name of the image. In case latest information is needed, the –refresh option can be used which would update the local database:

```
$cm image list 12 --cloud=kilo
+-------------------------------------+--------------------------------------+
| Attribute                           | Value                                |
+-------------------------------------+--------------------------------------+
| metadata__ramdisk_id                | None                                 |
| metadata__description               | None                                 |
| metadata__kernel_id                 | None                                 |
| id                                  | 12                                   |
| metadata__instance_type_ephemeral_gb | 0                                    |
| minRam                              | 0                                    |
| metadata__instance_type_swap        | 0                                    |
| metadata__instance_type_vcpus       | 1                                    |
| metadata__instance_type_rxtx_factor | 1.0                                  |
| progress                            | 100                                  |
| os_image_size                       | 1977483264                           |
| metadata__instance_type_flavorid    | 2                                    |
| metadata__instance_type_root_gb     | 20                                   |
| minDisk                             | 20                                   |
| created                             | 2015-05-23T20:45:51Z                 |
| updated                             | 2015-05-23T20:51:12Z                 |
| updated_at                          | 2015-11-11 00:29:55                  |
| created_at                          | 2015-11-11 00:29:55                  |
| metadata__instance_type_memory_mb   | 2048                                 |
| metadata__instance_type_id          | 5                                    |
| metadata__base_image_ref            | 6a6a3474-8194-44ac-9f56-70cb93207f21 |
| status                              | ACTIVE                               |
| metadata__network_allocated         | True                                 |
| uuid                                | a59833a2-60c9-47f0-b333-4e0bc071ac3a |
| metadata__image_state               | available                            |
| metadata__user_id                   | b13b62690e984c7586df1cdd2df07b5f     |
| metadata__owner_id                  | c713809dee494dccac34fcd02e012acb     |
| user                                | albert                               |
| metadata__instance_uuid             | f01633b1-76b0-47b5-915e-eaae4559ba60 |
| label                               | ZZZ                                  |
| name                                | ZZZ                                  |
| kind                                | image                                |
| cloud                               | kilo                                 |
| metadata__instance_type_name        | m1.small                             |
| metadata__image_location            | snapshot                             |
| metadata__image_type                | snapshot                             |
| project                             | undefined                            |
+-------------------------------------+--------------------------------------+
```

7.2.10 Network Command

The Network command provides an API that allows users to set up and define network connectivity and addressing in the cloud. Network command handles the creation and management of a virtual networking infrastructure, including networks, fixed & floating ips.

The manual page of the network command can be found at: *network <../man/man.html#network>*

..note:: We assume you have your default cloud set, via the default command:

```
    $ cm default cloud=kilo
```

List Floating IP Pools

To list the floating ip pools in your cloud network use:

```
$ cm network list floating pool
  +------------------+
  | floating_ip_pool |
  +------------------+
  | ext-net          |
  +------------------+
```

List Floating IP Addresses

To list the floating ip addresses in you cloud use:

```
$ cm network list floating ip
  +---------------+-----------------+------------------+-----------+----------------------------------
  | instance_name | floating_ip     | floating_ip_pool | fixed_ip  | floating_ip_id
  +---------------+-----------------+------------------+-----------+----------------------------------
  |               | 100.165.123.110 | ext-net          |           | 3e0915a9-f190-324d-8b56-4c2fd2a0
  | albert-004    | 100.165.123.111 | ext-net          | 10.0.2.10 | 58fbeca5-aad3-2f44-af23-0bb8ac60
  +---------------+-----------------+------------------+-----------+----------------------------------
```

To view the floating ip details for a particular instance, use:

```
$ cm network list floating ip --instance=albert-004
  +---------------+--------------------------------------+
  | name          | value                                |
  +---------------+--------------------------------------+
  | fixed_ip      | 10.0.2.10                            |
  | ip            | 100.165.123.111                      |
  | id            | 58fbeca5-aad3-2f44-af23-0bb8ac60dc89 |
  | instance_id   | a183b85f-2d4r-44b9-933f-64562380286f |
  | pool          | ext-net                              |
  | project       | fg478                                |
  | user          | albert                               |
  | instance_name | albert-004                           |
  | cloud         | kilo                                 |
  +---------------+--------------------------------------+
```

To view details of a particular floating ip address, use:

```
$ cm network list floating ip 100.165.123.111
  +---------------+--------------------------------------+
  | name          | value                                |
  +---------------+--------------------------------------+
  | fixed_ip      | 10.0.2.10                            |
  | ip            | 100.165.123.111                      |
  | id            | 58fbeca5-aad3-2f44-af23-0bb8ac60dc89 |
  | instance_id   | a183b85f-2d4r-44b9-933f-64562380286f |
  | pool          | ext-net                              |
  | project       | fg478                                |
  | user          | albert                               |
  | instance_name | albert-004                           |
```

```
  | cloud          | kilo                                  |
  +----------------+---------------------------------------+
```

Create Floating IP Addresses

To create a floating ip address under a floating pool, use:

```
$ cm network create floating ip --pool=ext-net
  Created new floating IP [100.165.123.112]

$ cm network list floating ip
  +---------------+-----------------+------------------+-----------+----------------------+-------------
  | instance_name | floating_ip     | floating_ip_pool | fixed_ip  | floating_ip_id
  +---------------+-----------------+------------------+-----------+----------------------+-------------
  |               | 100.165.123.110 | ext-net          |           | 3e0915a9-f190-324d-8b56-4c2fd2a0
  |               | 100.165.123.112 | ext-net          |           | 2cd915a9-f191-762d-2456-24dcd2a0
  | albert-004    | 100.165.123.111 | ext-net          | 10.0.2.10 | 58fbeca5-aad3-2f44-af23-0bb8ac60
  +---------------+-----------------+------------------+-----------+----------------------+-------------
```

Delete Floating IP Addresses

To delete a floating ip address, use:

```
$ cm network delete floating ip 100.165.123.112
  Floating IP [100.165.123.112] deleted successfully!

$ cm network list floating ip
  +---------------+-----------------+------------------+-----------+----------------------+-------------
  | instance_name | floating_ip     | floating_ip_pool | fixed_ip  | floating_ip_id
  +---------------+-----------------+------------------+-----------+----------------------+-------------
  |               | 100.165.123.110 | ext-net          |           | 3e0915a9-f190-324d-8b56-4c2fd2a0
  | albert-004    | 100.165.123.111 | ext-net          | 10.0.2.10 | 58fbeca5-aad3-2f44-af23-0bb8ac60
  +---------------+-----------------+------------------+-----------+----------------------+-------------
```

Associate Floating IP Address with an Instance

To automatically generate a floating ip address and associate it with an instance, use:

```
$ cm network associate floating ip --instance=albert-009
  Created and assigned Floating IP [100.165.123.113] to instance [albert-009].

$ cm network list floating ip
  +---------------+-----------------+------------------+-----------+----------------------+-------------
  | instance_name | floating_ip     | floating_ip_pool | fixed_ip  | floating_ip_id
  +---------------+-----------------+------------------+-----------+----------------------+-------------
  |               | 100.165.123.110 | ext-net          |           | 3e0915a9-f190-324d-8b56-4c2fd2a0
  | albert-004    | 100.165.123.111 | ext-net          | 10.0.2.10 | 58fbeca5-aad3-2f44-af23-0bb8ac60
  | albert-009    | 100.165.123.113 | ext-net          | 10.0.2.11 | 34fbeca5-aad3-4er5-ag21-34b8ac60
  +---------------+-----------------+------------------+-----------+----------------------+-------------
```

Alternatively, you can also specify the floating ip address that you want to associate with an instance:

```
$ cm network associate floating ip --instance=albert-008 100.165.123.112
  Associated Floating IP [100.165.123.112] to instance [albert-008].
```

```
$ cm network list floating ip
  +---------------+-----------------+------------------+-----------+----------------------
  | instance_name | floating_ip     | floating_ip_pool | fixed_ip  | floating_ip_id
  +---------------+-----------------+------------------+-----------+----------------------
  |               | 100.165.123.110 | ext-net          |           | 3e0915a9-f190-324d-8b56-4c2fd2a
  | albert-004    | 100.165.123.111 | ext-net          | 10.0.2.10 | 58fbeca5-aad3-2f44-af23-0bb8ac6
  | albert-008    | 100.165.123.112 | ext-net          | 10.0.2.12 | c45beca5-cd34-4e3d-4r34-34b8ac6
  | albert-009    | 100.165.123.113 | ext-net          | 10.0.2.11 | 34fbeca5-aad3-4er5-ag21-34b8ac6
  +---------------+-----------------+------------------+-----------+----------------------
```

Disassociate Floating IP Address from an Instance

To automatically detect the floating ip address associated with an instance & disassociate it from that instance, use:

```
$ cm network disassociate floating ip --instance=albert-009
  Disassociated Floating IP [100.165.123.113] from instance [albert-009].

$ cm network list floating ip
  +---------------+-----------------+------------------+-----------+----------------------
  | instance_name | floating_ip     | floating_ip_pool | fixed_ip  | floating_ip_id
  +---------------+-----------------+------------------+-----------+----------------------
  |               | 100.165.123.110 | ext-net          |           | 3e0915a9-f190-324d-8b56-4c2fd2a
  |               | 100.165.123.113 | ext-net          |           | 34fbeca5-aad3-4er5-ag21-34b8ac6
  | albert-004    | 100.165.123.111 | ext-net          | 10.0.2.10 | 58fbeca5-aad3-2f44-af23-0bb8ac6
  | albert-008    | 100.165.123.112 | ext-net          | 10.0.2.12 | c45beca5-cd34-4e3d-4r34-34b8ac6
  +---------------+-----------------+------------------+-----------+----------------------
```

Alternatively, you could also specify the floating ip address to dissociate:

```
$ cm network disassociate floating ip 100.165.123.113
  Disassociated Floating IP [100.165.123.113] from instance [albert-009].

$ cm network list floating ip
  +---------------+-----------------+------------------+-----------+----------------------
  | instance_name | floating_ip     | floating_ip_pool | fixed_ip  | floating_ip_id
  +---------------+-----------------+------------------+-----------+----------------------
  |               | 100.165.123.110 | ext-net          |           | 3e0915a9-f190-324d-8b56-4c2fd2a
  |               | 100.165.123.113 | ext-net          |           | 34fbeca5-aad3-4er5-ag21-34b8ac6
  | albert-004    | 100.165.123.111 | ext-net          | 10.0.2.10 | 58fbeca5-aad3-2f44-af23-0bb8ac6
  | albert-008    | 100.165.123.112 | ext-net          | 10.0.2.12 | c45beca5-cd34-4e3d-4r34-34b8ac6
  +---------------+-----------------+------------------+-----------+----------------------
```

Note: There are also a set of fixed-ip address operations you can perform, but you need to have admin privilidges in your account.

Some of the commands include:

Reserving a fixed ip address:

```
$ cm network reserve fixed ip 10.1.1.3
```

Unreserve a fixed ip address:

```
$ cm network unreserve fixed ip 10.1.1.3
```

Getting fixed ip address details:

```
$ cm network get fixed ip 10.1.1.3
```

7.2.11 Sync Command

The sync command provides an API that allows users to sync a local directory with a directory on any remote machine on the cloud. Sync command can be used to pull in data from the remote host, or send data from local machine to remote host.

The manual page of the network command can be found at: *sync <../man/man.html#sync>*

Sync file on local machine with remote machine on cloud

To sync a file from local machine to remote use:

```
$ cm sync put ubuntu_file.txt sync_dir
  Please enter putty private key(ppk) file path: ~/.ssh/id_rsa_ppk.ppk
  Passphrase for key "imported-openssh-key":
  ubuntu_file.txt           | 0 kB |   0.0 kB/s | ETA: 00:00:00 | 100%
  Successuly synced local and remote directories.
```

Sync file from remote machine on cloud to local machine

To sync a file from remote machine to local use:

```
$ cm sync get sync_dir/* ./cm_sync/
  Please enter putty private key(ppk) file path: ~/.ssh/id_rsa_ppk.ppk
  Passphrase for key "imported-openssh-key":
  my_text.txt               | 0 kB |   0.0 kB/s | ETA: 00:00:00 | 100%
  my_text_2.txt             | 0 kB |   0.0 kB/s | ETA: 00:00:00 | 100%
  ex1.txt                   | 0 kB |   0.0 kB/s | ETA: 00:00:00 | 100%
  ubuntu_file.txt           | 0 kB |   0.0 kB/s | ETA: 00:00:00 | 100%
  Successuly synced local and remote directories.
```

7.2.12 Limits Command

The manual page of the limits command can be found at: Limits

Accounts may be pre-configured with a set of thresholds(or limits) to manage capacity and prevent abuse of the system. Limits command gives a description of the limits set for each resource along with the total number of resources used.

limits list

To list the limits on a default project/tenant you can use:

```
$ cm limits list
  +-------------------------+-------+
  | Name                    | Value |
  +-------------------------+-------+
  | maxImageMeta            | 128   |
  | maxPersonality          | 5     |
  | maxPersonalitySize      | 10240 |
```

```
  | maxSecurityGroupRules    | 20    |
  | maxSecurityGroups        | 10    |
  | maxServerGroupMembers    | 10    |
  | maxServerGroups          | 10    |
  | maxServerMeta            | 128   |
  | maxTotalCores            | 20    |
  | maxTotalFloatingIps      | 10    |
  | maxTotalInstances        | 10    |
  | maxTotalKeypairs         | 100   |
  | maxTotalRAMSize          | 51200 |
  | totalCoresUsed           | 4     |
  | totalFloatingIpsUsed     | 0     |
  | totalInstancesUsed       | 4     |
  | totalRAMUsed             | 8192  |
  | totalSecurityGroupsUsed | 1     |
  | totalServerGroupsUsed    | 0     |
  +--------------------------+-------+
```

To export it in csv format, mention the format as csv:

```
$ cm limits list --format=csv
Name,Value
maxServerMeta,128
maxPersonality,5
totalServerGroupsUsed,0
maxImageMeta,128
maxPersonalitySize,10240
maxTotalRAMSize,51200
maxTotalKeypairs,100
maxSecurityGroupRules,20
maxServerGroups,10
totalCoresUsed,4
totalRAMUsed,8192
maxSecurityGroups,10
totalFloatingIpsUsed,0
totalInstancesUsed,4
totalSecurityGroupsUsed,1
maxTotalFloatingIps,10
maxTotalInstances,10
maxTotalCores,20
maxServerGroupMembers,10
```

7.2.13 Quota Command

Many clouds have some kind of quota limitations on how many ip addresses one can obtain, or how many cores a user can have. To get an overview of the quotas set for a user in a project we are providing a quota command.

The manual page of the quota command can be found at: Quota

quota list

To list the quota limit on a default project/tenant you can use:

```
$ cm quota list
  +-----------------------------+-------+
  | Quota                       | Limit |
```

```
  +-----------------------------+-------+
  | fixed_ips                   | -1    |
  | floating_ips                | 10    |
  | instances                   | 10    |
  | security_groups             | 10    |
  | server_group_members        | 10    |
  | server_groups               | 10    |
  | key_pairs                   | 100   |
  | injected_file_content_bytes | 10240 |
  | metadata_items              | 128   |
  | cores                       | 20    |
  | security_group_rules        | 20    |
  | injected_file_path_bytes    | 255   |
  | injected_files              | 5     |
  | ram                         | 51200 |
  +-----------------------------+-------+
```

To export it in csv format,:

```
$ cm quota list --format=csv
Quota,Limit
instances,10
cores,20
ram,51200
floating_ips,10
fixed_ips,-1
metadata_items,128
injected_files,5
injected_file_content_bytes,10240
injected_file_path_bytes,255
key_pairs,100
security_groups,10
security_group_rules,20
server_groups,10
server_group_members,10
```

7.2.14 Usage Command

The manual page of the quota command can be found at: Usage

This shows the resource list for a particular time frame. By default it will show the resource list for the past one month.

list

To list the list information:

```
$cm list
Usage from 2015-09-24 to 2015-10-23:
+-----------+----------------+-------------+-----------------+
|  Servers  |  RAM MB-Hours  |  CPU Hours  |  Disk GB-Hours  |
+-----------+----------------+-------------+-----------------+
|  42       |  6275181.11    |  3064.05    |  61281.07       |
+-----------+----------------+-------------+-----------------+
```

7.3 HPC Commands

7.3.1 Hpc Command

High Performance Computing(HPC) allows to solve large complex problems in engineering, science and business using applications that require very high compute power and amplified bandwidth. The cloudmesh hpc command helps to easily manage hpc clusters.

The manual page of the hpc command can be found at: Hpc

Before we get started, we can set the default hpc cluster or use the –cluster option. To set the default hpc cluster:

```
$ cm default cluster=comet
set in defaults cluster=comet. ok.
```

hpc info

Returns the state of partitions and nodes on the hpc cluster:

```
$ cm hpc info
  +---------+-----------+-------+------------+-------+-------+---------------------------+----------
  | cluster | partition | avail | timelimit  | nodes | state | nodelist                  | updated
  +---------+-----------+-------+------------+-------+-------+---------------------------+----------
  | india   | xxxxx     | up    | 3-00:00:00 | 8     | idle  | b[009-016]                | 2015-11-2
  | india   | yyyyy     | up    | 3-00:00:00 | 12    | idle  | d[001-012]                | 2015-11-2
  | india   | zzzzz     | up    | 3-00:00:00 | 16    | idle  | i[81-84,86-89,91-95,97-99] | 2015-11-2
  +---------+-----------+-------+------------+-------+-------+---------------------------+----------
```

hpc queue

Reports the state of jobs or job sets:

```
$ cm hpc queue
+---------+---------+--------------+-------------------+-----------+----+------------+-------+------
| cluster | jobid   | partition    | name              | user      | st | time       | nodes | nodeli
+---------+---------+--------------+-------------------+-----------+----+------------+-------+------
| india   | 1205397 | gpu-shared   | xxx               | x_user    | PD | 0:00       | 1     |
| india   | 1267689 | compute      | yyy               | y_user    | PD | 0:00       | 1     |
| india   | 1267690 | compute      | zzz               | y_user    | PD | 0:00       | 8     |
| india   | 1267691 | compute      | lll               | y_user    | PD | 0:00       | 3     |
| india   | 1267693 | compute      | mmm               | y_user    | PD | 0:00       | 1     |
| india   | 1295159 | gpu          | nnnnnnn           | z_user    | CG | 1-00:00:03 | 1     | xxxxx
| india   | 1304301 | compute      | ooooooooooo       | y_user    | R  | 23:38:55   | 8     | yy-04-
+---------+---------+--------------+-------------------+-----------+----+------------+-------+------
```

To view the state of a specific job, use the –job=NAME option, where NAME can be the job id or the job name

```
$ cm hpc queue --job=6
+---------+-------+-----------+-------------+-----------+----+------+-------+----------+-------------
| cluster | jobid | partition | name        | user      | st | time | nodes | nodelist | updated
+---------+-------+-----------+-------------+-----------+----+------+-------+----------+-------------
| india   | 6     | xxxxx     | somethin.sh | xxxxxxxxx | PD | 0:00 | 1     |          | 2015-11-29 1
+---------+-------+-----------+-------------+-----------+----+------+-------+----------+-------------
```

hpc status

Similar to hpc queue where the status of job(s) can be viewed.

Experiment management

Often it is the case that you may want to rerun your script multiple times with potentially different parameters. We are working towards simplifying this mechanism for parameter studies. At this time we implemented the ability to run a simple shell command repeatedly.

For this we provide a simple experiment abstraction. An experiment is created automatically once you run an hpc command. The experiment is placed in an output directory that can be defined within the cloudmesh.yaml file. By default it will be the home directory ~/experiment. In this experiment we create numbered sub directories for consecutive execution of the experiment run.

To run an experiment (in this case just a shell command *uname*) you can use the run command:

hpc run uname –cluster=india

It will create on the cluster india a new experiment directory by increasing an experiment number, put the batch script, run the command, and put the output into this directory.

One can also transfer a script for the experiment, for example:

hpc run <script_path> –cluster=india

To list experiments that have been previously run you can use the command:

```
hpc run list
```

To list the files in a particular experiment you can use the experiment number:

```
hpc run list 11
```

Now you will see what the experiment script has created and you will be able to fr example view the output of the script:

```
hpc run output 11
```

To delete an experiment you can say:

```
hpc run rm 11
```

However, be careful as deleting it will permanently delete the file. To delete all experiments (be extra careful) you can just omit the number:

```
hpc run rm
```

In future we will provide the ability to add custom scripts.

hpc delete

If for any reason, you need to kill a job that you have submitted, use the delete command with the job name or the job id:

```
$ hpc delete --job=1463
Job 1463 killed successfully
```

To delete all jobs from a group:

```
$ hpc delete all
All jobs for group test killed successfully
```

The above command will delete all active jobs from the default group. You can also use the –group to specify a group of your choice.

7.4 Comet Commands

7.4.1 Comet Command

The manual page of the group command can be found at: comet

Cloudmesh has the ability to manage easily multiple clouds. ONe such cloud is comet. comet cloud is special as it allows the management of virtual clusters. It is intended for advanced users that can manage their own cluster environment.

Configuration

Configure the comet section in ~/.cloudmesh/cloudmesh.yaml file first. auth_provider could be userpass or apikey. When specified, the corresponding credential is needed. Please communicate with comet admins to get the username/password or api key and secret assigned.:

```
comet:
    auth_provider: apikey
    userpass:
        username: TBD
        password: TBD
    apikey:
        api_key: KEYSTRING
        api_secret: SECRETSTRING
```

Reference Guide

Next, we include a small set of useful examples to manage comet virtual cluster using cloudmesh client.

comet list

All the current default values can by listed with –all option:

```
$ comet ll
+--------+-----------+-------+-------+
| user   | cloud     | name  | value |
+--------+-----------+-------+-------+
| albert | chameleon | image | abc   |
| albert | general   | cloud | azure |
| albert | general   | image | zyx   |
+--------+-----------+-------+-------+
```

You can also add a –cloud=CLOUD option to see the defaults set for a cloud:

```
$ comet list
+--------+-----------+-------+-------+
| user   | cloud     | name  | value |
+--------+-----------+-------+-------+
| albert | chameleon | image | abc   |
+--------+-----------+-------+-------+
```

Options

```
--user=USER           user name
--name=NAMES          Names of the vcluster
--start=TIME_START    Start time of the vcluster, in
                      YYYY/MM/DD HH:MM:SS format.
                      [default: 1901-01-01]
--end=TIME_END        End time of the vcluster, in YYYY/MM/DD
                      HH:MM:SS format. In addition a duratio
                      can be specified if the + sign is the
                      first sig The duration will than be
                      added to the start time.
                      [default: 2100-12-31]
--project=PROJECT     project id
--host=HOST           host name
--description=DESCRIPTION  description summary of the vcluster
--file=FILE           Adding multiple vclusters from one file
--format=FORMAT       Format is either table, json, yaml,
                      csv, rest
                      [default: table]
```

Arguments

```
    FILENAME  the file to open in the cwd if . is
              specified. If file in in cwd
              you must specify it with ./FILENAME

Opens the given URL in a browser window.
```

comet tunnel

comet configuration

comet logon

comet docs

comet status

TBD

comet info

TBD:

```
comet info [--user=USER]
             [--project=PROJECT]
             [--format=FORMAT]
```

comet_cluster

```
comet cluster [ID][--name=NAMES]
             [--user=USER]
             [--project=PROJECT]
             [--hosts=HOSTS]
             [--start=TIME_START]
             [--end=TIME_END]
             [--hosts=HOSTS]
             [--format=FORMAT]
```

comet computeset

```
comet computeset [COMPUTESETID]
```

comet start and stop

```
comet start ID
```

```
comet stop ID
```

comet power

```
comet power (on|off|reboot|reset|shutdown) CLUSTERID PARAM
```

comet delete

```
comet delete [all]
               [--user=USER]
               [--project=PROJECT]
               [--name=NAMES]
               [--hosts=HOSTS]
               [--start=TIME_START]
               [--end=TIME_END]
               [--host=HOST]
comet delete --file=FILE
```

comet update

```
comet update [--name=NAMES]
               [--hosts=HOSTS]
               [--start=TIME_START]
               [--end=TIME_END]
```

comet add

```
comet add [--user=USER]
            [--project=PROJECT]
            [--host=HOST]
            [--description=DESCRIPTION]
            [--start=TIME_START]
            [--end=TIME_END]
            NAME
comet add --file=FILENAME
```

7.5 Proposed Commands

7.5.1 Reservation Command

Warning: This command is experimental and is not yet fully integrated. It only stores reservations, but does not act upon them.

One of the features of cloudmesh is to build a mesh of resources and services. In some cases we which to reserve resources that allow reservations. However we also may want to use a resource till a particular time frame and release it. For this cases it is practical to provide the concept of a reservation. A simple reservation is named and contains a start and end point. We currently store named virtual machines into a reservation for named clouds. Reservations are similar to groups just that they have a time frame associated with them. A timeless reservation is like a group.

The manual page of the key command can be found at: reservation

Adding a reservation

Please note that you have to escape the whitespaces with '\' for commmand line arguments such as '–start', '–end'.

```
$ cm reservation add --name=test3 --start='10/31/1988\ at\ 8:09\ pm' --end='10/21/2015\ at\ 9:00\ pm
Reservation test3 added successfully
info. OK.
```

List Reservation

```
$ cm reservation list
+----+-------+-----------------+-----------------+--------+-----------+---------+-------------+-----
| id | name  | start_time      | end_time        | user   | project   | hosts   | description | cloud
+----+-------+-----------------+-----------------+--------+-----------+---------+-------------+-----
| 1  | test3 | 10-31-1988 20:9 | 10-21-2015 21:0 | albert | cloudmesh | host001 | desc        | comet
+----+-------+-----------------+-----------------+--------+-----------+---------+-------------+-----
```

Update Reservation

Please note that you have to escape the whitespaces with '\' for command line arguments such as '–start', '–end'.

```
$ cm reservation update --name=test3 --project=cloudnauts
Reservation test3 updated successfully
info. OK.
```

Verify by listing:

```
$ cm reservation list
+----+-------+---------------+----------------+--------+------------+---------+-------------+-------
| id | name  | start_time    | end_time       | user   | project    | hosts   | description | cloud
+----+-------+---------------+----------------+--------+------------+---------+-------------+-------
| 1  | test3 | 1-1-1901 19:30 | 12-31-2021 0:0 | albert | cloudnauts | host001 | desc        | comet
+----+-------+---------------+----------------+--------+------------+---------+-------------+-------
```

Delete Reservation

```
$ cm reservation delete --name=test3
info. OK.
```

Verify by listing:

```
$ cm reservation list
None
info. OK.
```

7.5.2 Inventory Command

The manual page of the key command can be found at: Nova

Todo

reformat the inventory section to be a real manual.

Examples:

```
cm inventory add x[0-3] --service=openstack

    adds hosts x0, x1, x2, x3 and puts the string
    openstack into the service column

cm lists

    lists the repository

cm x[3-4] set temperature to 32

    sets for the resources x3, x4 the value of the
    temperature to 32

cm x[7-8] set ip 128.0.0.[0-1]

    sets the value of x7 to 128.0.0.0
    sets the value of x8 to 128.0.0.1

cm clone x[5-6] from x3

    clones the values for x5, x6 from x3
```

CHAPTER 8

Commands

8.1 banner

Command - banner:

```
Usage:
    banner [-c CHAR] [-n WIDTH] [-i INDENT] [-r COLOR] TEXT

Arguments:
    TEXT   The text message from which to create the banner
    CHAR   The character for the frame.
    WIDTH  Width of the banner
    INDENT indentation of the banner
    COLOR  the color

Options:
    -c CHAR   The character for the frame. [default: #]
    -n WIDTH  The width of the banner. [default: 70]
    -i INDENT  The width of the banner. [default: 0]
    -r COLOR  The color of the banner. [default: BLACK]

Prints a banner form a one line text message.
```

8.2 check

Command - check:

```
Usage:
    check --cloud=CLOUD
    check

    checks some elementary setting for cloudmesh

Options:
   --format=FORMAT  the output format [default: table]
   --cloud=CLOUD    the cloud name

Examples:
    cm check
    cm check --cloud=kilo
```

8.3 clear

Command - clear:

```
Usage:
    clear

Clears the screen.
```

8.4 cloud

Command - cloud:

```
Usage:
    cloud list [--cloud=CLOUD] [--format=FORMAT]
    cloud logon CLOUD
    cloud logout CLOUD
    cloud activate CLOUD
    cloud deactivate CLOUD
    cloud info CLOUD

managing the admins test test test test

Arguments:
  KEY    the name of the admin
  VALUE  the value to set the key to

Options:
   --cloud=CLOUD    the name of the cloud
   --format=FORMAT  the output format [default: table]

Description:
   Cloudmesh contains a cloudmesh.yaml file that contains
   templates for multiple clouds that you may or may not have
   access to. Hence it is useful to activate and deactivate clouds
   you like to use in other commands.

   To activate a cloud a user can simply use the activate
   command followed by the name of the cloud to be
   activated. To find out which clouds are available you can
   use the list command that will provide you with some
   basic information. As default it will print a table. Thus
   the commands       cloud activate india
     cloud deactivate aws

   Will result in

      +----------------------+--------+-------------------+
      | Cloud name           | Active | Type              |
      +----------------------+--------+-------------------+
      | india                | True   | Openstack         |
      +----------------------+--------+-------------------+
      | aws                  | False  | AWS               |
      +----------------------+--------+-------------------+

   To get ore information about the cloud you can use the command
```

```
      cloud info CLOUD

   It will call internally also the command uses in register

See also:
   register
```

8.5 cluster

Command - cluster:

```
Usage:
    cluster list [--format=FORMAT]
    cluster list NAME
                 [--format=FORMAT]
                 [--column=COLUMN]
                 [--detail]
    cluster create NAME
                   [--count=COUNT]
                   [--login=USERNAME]
                   [--cloud=CLOUD]
                   [--image=IMAGE]
                   [--flavor=FLAVOR]
                   [--add]
    cluster delete NAME

Description:
    with the help of the cluster command you can create a number
    of virtual machines that are integrated in a named virtual cluster.
    You will be able to login between the nodes of the virtual cluster
    while using public keys.

Examples:
    cluster list
        list the clusters

    cluster create NAME --count=COUNT --login=USERNAME [options...]
        Start a cluster of VMs, and each of them can log into each other.
        CAUTION: you should specify defaults before using this command:
        1. select cloud to work on, e.g. cloud select kilo
             default cloud=kilo
        2. test if you can create a single VM on the cloud to see if
           everything is set up
        3. set the default key to start VMs, e.g. key default [USERNAME-key]
        5. set image of VMs, e.g. default image
        6. set flavor of VMs, e.g. default flavor
        7. Make sure to use a new unused group name

    cluster list NAME
        show the detailed information about the cluster VMs

    cluster delete NAME
        remove the cluster and its VMs

Arguments:
    NAME              cluster name or group name
```

```
Options:
    --count=COUNT     give the number of VMs to add into the cluster
    --login=USERNAME  give a login name for the VMs, e.g. ubuntu
    --cloud=CLOUD     give a cloud to work on
    --flavor=FLAVOR   give the name of the flavor or flavor id
    --image=IMAGE     give the name of the image or image id
    --add             if a group exists and there are VMs in it
                      additional vms will be added to this cluster and the
                      keys will be added to each other so one can login between
                      them
    FORMAT            output format: table, json, csv
    COLUMN            customize what information to display, for example:
                      --column=status,addresses prints the columns status
                      and addresses
    --detail          for table print format, a brief version
                      is used as default, use this flag to print
                      detailed table
```

8.6 color

Command - color:

```
Usage:
    color FLAG

Arguments:

    FLAG    color mode flag ON/OFF

Description:

    Global switch for the console color mode.
    One can switch the color mode on/off with
        cm color mode ON
        cm color mode OFF

    By default, the color mode is ON

Examples:
    color mode ON
    color mode OFF
```

8.7 comet

Command - comet:

```
Usage:
   comet status
   comet tunnel start
   comet tunnel stop
   comet tunnel status
   comet logon
   comet logoff
   comet ll [CLUSTERID] [--format=FORMAT]
```

```
   comet docs
   comet info [--user=USER]
              [--project=PROJECT]
              [--format=FORMAT]
   comet cluster [CLUSTERID][--name=NAMES]
              [--user=USER]
              [--project=PROJECT]
              [--hosts=HOSTS]
              [--start=TIME_START]
              [--end=TIME_END]
              [--hosts=HOSTS]
              [--format=FORMAT]
   comet computeset [COMPUTESETID]
   comet start ID
   comet stop ID
   comet power (on|off|reboot|reset|shutdown) CLUSTERID [NODESPARAM]
   comet console CLUSTERID [COMPUTENODEID]
   comet delete [all]
                [--user=USER]
                [--project=PROJECT]
                [--name=NAMES]
                [--hosts=HOSTS]
                [--start=TIME_START]
                [--end=TIME_END]
                [--host=HOST]
   comet delete --file=FILE
   comet update [--name=NAMES]
                [--hosts=HOSTS]
                [--start=TIME_START]
                [--end=TIME_END]
   comet add [--user=USER]
             [--project=PROJECT]
             [--host=HOST]
             [--description=DESCRIPTION]
             [--start=TIME_START]
             [--end=TIME_END]
             NAME
   comet add --file=FILENAME

Options:
    --user=USER           user name
    --name=NAMES          Names of the vcluster
    --start=TIME_START    Start time of the vcluster, in
                          YYYY/MM/DD HH:MM:SS format.
                          [default: 1901-01-01]
    --end=TIME_END        End time of the vcluster, in YYYY/MM/DD
                          HH:MM:SS format. In addition a duratio
                          can be specified if the + sign is the
                          first sig The duration will than be
                          added to the start time.
                          [default: 2100-12-31]
    --project=PROJECT     project id
    --host=HOST           host name
    --description=DESCRIPTION  description summary of the vcluster
    --file=FILE           Adding multiple vclusters from one file
    --format=FORMAT       Format is either table, json, yaml,
                          csv, rest
                          [default: table]
```

```
Arguments:
    FILENAME  the file to open in the cwd if . is
              specified. If file in in cwd
              you must specify it with ./FILENAME

Opens the given URL in a browser window.
```

8.8 context

Command - context:

```
Usage:
    context

Description:
    Lists the context variables and their values
```

8.9 debug

Command - debug:

```
Usage:
    debug on
    debug off
    debug list

    switches on and off the debug messages
```

8.10 default

Command - default:

```
  Usage:
      default
      default list [--cloud=CLOUD] [--format=FORMAT] [--all]
      default delete KEY [--cloud=CLOUD]
      default KEY [--cloud=CLOUD]
      default KEY=VALUE [--cloud=CLOUD]

  Arguments:
    KEY    the name of the default
    VALUE  the value to set the key to

  Options:
     --cloud=CLOUD    the name of the cloud
     --format=FORMAT  the output format. Values include
                      table, json, csv, yaml. [default: table]
     --all            lists all the default values

Description:
    Cloudmesh has the ability to manage easily multiple
```

```
    clouds. One of the key concepts to manage multiple clouds
    is to use defaults for the cloud, the images, flavors,
    and other values. The default command is used to manage
    such default values. These defaults are used in other commands
    if they are not overwritten by a command parameter.

    The current default values can by listed with

        default list --all

    Via the default command you can list, set, get and delete
    default values. You can list the defaults with

       default list

    A default can be set with

        default KEY=VALUE

    To look up a default value you can say

        default KEY

    A default can be deleted with

        default delete KEY

    To be specific to a cloud you can specify the name of the
    cloud with the --cloud=CLOUD option. The list command can
    print the information in various formats iv specified.

Examples:
    default
        lists the default for the current default cloud

    default list --all
        lists all default values

    default list --cloud=kilo
        lists the defaults for the cloud with the name kilo

    default image=xyz
        sets the default image for the default cloud to xyz

    default image=abc --cloud=kilo
        sets the default image for the cloud kilo to xyz

    default image
        list the default image of the default cloud

    default image --cloud=kilo
        list the default image of the cloud kilo

    default delete image
        deletes the value for the default image in the
        default cloud

    default delete image --cloud=kilo
```

```
        deletes the value for the default image in the
        cloud kilo
```

8.11 echo

Command - echo:

```
Usage:
    echo  [-r COLOR] TEXT

Arguments:
    TEXT   The text message to print
    COLOR  the color

Options:
    -r COLOR  The color of the text. [default: BLACK]

Prints a text in the given color
```

8.12 EOF

Command - EOF:

```
Usage:
    EOF

Description:
    Command to the shell to terminate reading a script.
```

8.13 exec

Command - exec:

```
Usage:
   exec FILENAME

executes the commands in the file. See also the script command.

Arguments:
  FILENAME   The name of the file
```

8.14 flavor

Command - flavor:

```
Usage:
    flavor refresh [--cloud=CLOUD] [-v]
    flavor list [ID] [--cloud=CLOUD] [--format=FORMAT] [--refresh] [-v]

    This lists out the flavors present for a cloud
```

```
Options:
   --format=FORMAT  the output format [default: table]
   --cloud=CLOUD    the cloud name
   --refresh        refreshes the data before displaying it
                    from the cloud

Examples:
    cm flavor refresh
    cm flavor list
    cm flavor list --format=csv
    cm flavor show 58c9552c-8d93-42c0-9dea-5f48d90a3188 --refresh
```

8.15 group

Command - group:

```
Usage:
    group add NAME [--type=TYPE] [--cloud=CLOUD] [--id=IDs]
    group list [--cloud=CLOUD] [--format=FORMAT] [NAME]
    group delete NAME [--cloud=CLOUD]
    group remove [--cloud=CLOUD] --name=NAME --id=ID
    group copy FROM TO
    group merge GROUPA GROUPB MERGEDGROUP

manage the groups

Arguments:

    NAME         name of a group
    FROM         name of a group
    TO           name of a group
    GROUPA       name of a group
    GROUPB       name of a group
    MERGEDGROUP  name of a group

Options:
    --cloud=CLOUD    the name of the cloud
    --format=FORMAT  the output format
    --type=TYPE     the resource type
    --name=NAME      the name of the group


Description:

    Todo: design parameters that are useful and match
    description
    Todo: discuss and propose command

    cloudmesh can manage groups of resources and cloud related
    objects. As it would be cumbersome to for example delete
    many virtual machines or delete VMs that are in the same
    group, but are running in different clouds.

    Hence it is possible to add a virtual machine to a
    specific group. The group name to be added to can be set
    as a default. This way all subsequent commands use this
```

```
    default group. It can also be set via a command parameter.
    Another convenience function is that the group command can
    use the last used virtual machine. If a vm is started it
    will be automatically added to the default group if it is set.

    The delete command has an optional cloud parameter so that
    deletion of vms of a partial group by cloud can be
    achieved.

    If finer grained deletion is needed, it can be achieved
    with the delete command that supports deletion by name

    It is also possible to remove a VM from the group using the
    remove command, by supplying the ID

Example:
    default group mygroup

    group add --type=vm --id=albert-[001-003]
        adds the vms with teh given name using the Parameter
        see base

    group add --type=vm
     adds the last vm to the group

    group delete --name=mygroup
        deletes all objects in the group
```

8.16 h

Command - h:

```
Usage:
    history
    history list
    history last
    history ID
```

8.17 help

Command - help:

```
Usage:
    help
    help COMMAND

Description:
    List available commands with "help" or detailed help with
    "help COMMAND".
```

8.18 history

Command - history:

```
Usage:
    history
    history list
    history last
    history ID
```

8.19 hpc

Command - hpc:

```
Usage:
    hpc queue [--job=NAME][--cluster=CLUSTER][--format=FORMAT]
    hpc info [--cluster=CLUSTER][--format=FORMAT]
    hpc run list [ID] [--cluster=CLUSTER]
    hpc run output [ID] [--cluster=CLUSTER]
    hpc run rm [ID] [--cluster=CLUSTER]
    hpc run SCRIPT [--queue=QUEUE] [--t=TIME] [--N=nodes] [--name=NAME] [--cluster=CLUSTER][--dir=DI
    hpc delete --job=NAME [--cluster=CLUSTER][--group=GROUP]
    hpc delete all [--cluster=CLUSTER][--group=GROUP][--format=FORMAT]
    hpc status [--job=name] [--cluster=CLUSTER] [--group=GROUP]
    hpc test --cluster=CLUSTER [--time=SECONDS]

Options:
   --format=FORMAT  the output format [default: table]

Examples:

    Special notes

       if the group is specified only jobs from that group are
       considered. Otherwise the default group is used. If the
       group is set to None, all groups are used.

    cm hpc queue
        lists the details of the queues of the hpc cluster

    cm hpc queue --job=NAME
        lists the details of the job in the queue of the hpc cluster

    cm hpc info
        lists the details of the hpc cluster

    cm hpc run SCRIPT
        submits the script to the cluster. The script will be
        copied prior to execution into the home directory on the
        remote machine. If a DIR is specified it will be copied
        into that dir.
        The name of the script is either specified in the script
        itself, or if not the default naming scheme of
        cloudmesh is used using the same index incremented name
        as in vms fro clouds: cloudmes husername-index
```

```
    cm hpc delete all
        kills all jobs on the default hpc group

    cm hpc delete --job=NAME
        kills a job with a given name or job id

    cm default cluster=NAME
        sets the default hpc cluster

    cm hpc status
        returns the status of all jobs

    cm hpc status job=ID
        returns the status of the named job

    cm hpc test --cluster=CLUSTER --time=SECONDS
        submits a simple test job to the named cluster and returns
        if the job could be successfully executed. This is a
        blocking call and may take a long time to complete
        dependent on if the queuing system of that cluster is
        busy. It will only use one node/core and print the message

        #CLOUDMESH: Test ok

        in that is being looked for to identify if the test is
        successful. If time is used, the job is terminated
        after the time is elapsed.

Examples:
    cm hpc queue
    cm hpc queue --job=xxx
    cm hpc info
    cm hpc delete --job=6
    cm hpc delete all
    cm hpc status
    cm hpc status --job=6
    cm hpc run uname
    cm hpc run ~/test.sh --cluster=india
```

8.20 image

Command - image:

```
Usage:
    image refresh [--cloud=CLOUD]
    image list [ID] [--cloud=CLOUD] [--format=FORMAT] [--refresh]

    This lists out the images present for a cloud

Options:
   --format=FORMAT  the output format [default: table]
   --cloud=CLOUD    the cloud name
   --refresh        live data taken from the cloud

Examples:
    cm image refresh
```

```
    cm image list
    cm image list --format=csv
    cm image list 58c9552c-8d93-42c0-9dea-5f48d90a3188 --refresh
```

8.21 inventory

Command - inventory:

```
Usage:
    inventory add NAMES [--label=LABEL]
                        [--service=SERVICES]
                        [--project=PROJECT]
                        [--owners=OWNERS]
                        [--comment=COMMENT]
                        [--cluster=CLUSTER]
                        [--ip=IP]
    inventory set NAMES for ATTRIBUTE to VALUES
    inventory delete NAMES
    inventory clone NAMES from SOURCE
    inventory list [NAMES] [--format=FORMAT] [--columns=COLUMNS]
    inventory info

Arguments:

  NAMES     Name of the resources (example i[10-20])

  FORMAT    The format of the output is either txt,
            yaml, dict, table [default: table].

  OWNERS    a comma separated list of owners for this resource

  LABEL     a unique label for this resource

  SERVICE   a string that identifies the service

  PROJECT   a string that identifies the project

  SOURCE    a single host name to clone from

  COMMENT   a comment

Options:

   -v       verbose mode

Description:

      add -- adds a resource to the resource inventory

      list -- lists the resources in the given format

      delete -- deletes objects from the table

      clone -- copies the content of an existing object
               and creates new once with it
```

```
      set   -- sets for the specified objects the attribute
               to the given value or values. If multiple values
               are used the values are assigned to the and
               objects in order. See examples

      map   -- allows to set attibutes on a set of objects
               with a set of values

Examples:

  cm inventory add x[0-3] --service=openstack

      adds hosts x0, x1, x2, x3 and puts the string
      openstack into the service column

  cm lists

      lists the repository

  cm x[3-4] set temperature to 32

      sets for the resources x3, x4 the value of the
      temperature to 32

  cm x[7-8] set ip 128.0.0.[0-1]

      sets the value of x7 to 128.0.0.0
      sets the value of x8 to 128.0.0.1

  cm clone x[5-6] from x3

      clones the values for x5, x6 from x3
```

8.22 key

Command - key:

```
Usage:
  key  -h | --help
  key list [--source=db] [--format=FORMAT]
  key list --source=cloudmesh [--format=FORMAT]
  key list --source=ssh [--dir=DIR] [--format=FORMAT]
  key list --source=git [--format=FORMAT] [--username=USERNAME]
  key add --git [--name=KEYNAME] FILENAME
  key add --ssh [--name=KEYNAME]
  key add [--name=KEYNAME] FILENAME
  key get NAME
  key default [KEYNAME | --select]
  key delete (KEYNAME | --select | --all) [-f]
  key upload KEYNAME
                  [--cloud=CLOUD]
                  [--name=NAME_ON_CLOUD]
  key map [--cloud=CLOUD]

Manages the keys
```

```
Arguments:

  SOURCE         db, ssh, all
  KEYNAME        The name of a key
  FORMAT         The format of the output (table, json, yaml)
  FILENAME       The filename with full path in which the key
                 is located
  NAME_ON_CLOUD  Typically the name of the keypair on the cloud.

Options:

   --dir=DIR                     the directory with keys [default: ~/.ssh]
   --format=FORMAT               the format of the output [default: table]
   --source=SOURCE               the source for the keys [default: db]
   --username=USERNAME           the source for the keys [default: none]
   --name=KEYNAME                The name of a key
   --all                         delete all keys
   --name_on_cloud=NAME_ON_CLOUD Typically the name of the keypair on the cloud.

Description:

key list --source=git  [--username=USERNAME]

   lists all keys in git for the specified user. If the
   name is not specified it is read from cloudmesh.yaml

key list --source=ssh  [--dir=DIR] [--format=FORMAT]

   lists all keys in the directory. If the directory is not
   specified the default will be ~/.ssh

key list --source=cloudmesh  [--dir=DIR] [--format=FORMAT]

   lists all keys in cloudmesh.yaml file in the specified directory.
    dir is by default ~/.cloudmesh

key list [--format=FORMAT]

    list the keys in teh giiven format: json, yaml,
    table. table is default

key list

     Prints list of keys. NAME of the key can be specified


key add [--name=keyname] FILENAME

    adds the key specifid by the filename to the key
    database

key get NAME

    Retrieves the key indicated by the NAME parameter from database
    and prints its fingerprint.

key default [NAME]
```

```
     Used to set a key from the key-list as the default key
     if NAME is given. Otherwise print the current default
     key

key delete NAME

     deletes a key. In yaml mode it can delete only key that
     are not saved in the database

key rename NAME NEW

     renames the key from NAME to NEW.
```

8.23 launcher

Command - launcher:

```
  Usage:
      launcher list [--cloud=CLOUD] [--format=FORMAT] [--all]
      launcher delete KEY [--cloud=CLOUD]
      launcher run
      launcher resume
      launcher suspend
      launcher details
      launcher clear
      launcher refresh

  Arguments:

    KEY    the name of the launcher

  Options:

     --cloud=CLOUD    the name of the cloud
     --format=FORMAT  the output format [launcher: table]
     --all            lists all the launcher values

Description:

Launcher is a command line tool to test the portal launch functionalities through command

The current launcher values can by listed with --all option:(
if you have a launcher cloud specified. You can also add a
cloud parameter to apply the command to a specific cloud)

       launcher list

    A launcher can be deleted with

        launcher delete KEY


Examples:
    launcher list --all
    launcher list --cloud=general
    launcher delete <KEY>
```

8.24 limits

Command - limits:

```
Usage:
    limits list [--cloud=CLOUD] [--tenant=TENANT] [--format=FORMAT]

    Current list data with limits on a selected project/tenant.
    The --tenant option can be used by admin only

Options:
   --format=FORMAT  the output format [default: table]
   --cloud=CLOUD    the cloud name
   --tenant=TENANT  the tenant name

Examples:
    cm limits list
    cm limits list --cloud=kilo --format=csv
```

8.25 list

Command - list:

```
Usage:
    list [--cloud=CLOUD] [--format=FORMAT] [--user=USER] [--tenant=TENANT] default
    list [--cloud=CLOUD] [--format=FORMAT] [--user=USER] [--tenant=TENANT] vm
    list [--cloud=CLOUD] [--format=FORMAT] [--user=USER] [--tenant=TENANT] flavor
    list [--cloud=CLOUD] [--format=FORMAT] [--user=USER] [--tenant=TENANT] image

List the items stored in the database

Options:
    --cloud=CLOUD    the name of the cloud
    --format=FORMAT  the output format
    --tenant=TENANT     Name of the tenant, e.g. fg82.

Description:
    List command prints the values stored in the database
    for [default/vm/flavor/image].
    Result can be filtered based on the cloud, user & tenant arguments.
    If these arguments are not specified, it reads the default

Examples:
    $ list --cloud india default
    $ list --cloud india --format table flavor
    $ list --cloud india --user albert --tenant fg82 flavor
```

8.26 loglevel

Command - loglevel:

```
    Usage:
        loglevel set MODE [--cloud=CLOUD]
        loglevel get [--cloud=CLOUD]
```

```
        loglevel save [--cloud=CLOUD]

    Arguments:
        MODE    log level mode [DEBUG/INFO/WARNING/CRITICAL/ERROR]

    Options:
        --cloud=CLOUD    the name of the cloud

Description:
    loglevel command sets the default logging level
    for a cloud.

Examples:
    loglevel set DEBUG --cloud=kilo
        sets the default log level to DEBUG for kilo.

    loglevel get --cloud=kilo
        retreives the default log level for kilo cloud.

    loglevel save --cloud=kilo
        saves the log level preference to the db & yaml file.
```

8.27 man

Command - man:

```
Usage:
       man COMMAND
       man [--noheader]

Options:
       --norule   no rst header

Arguments:
       COMMAND   the command to be printed

Description:
    man
        Prints out the help pages

    man COMMAND
        Prints out the help page for a specific command
```

8.28 network

Command - network:

```
Usage:
    network get fixed [ip] [--cloud=CLOUD] FIXED_IP
    network get floating [ip] [--cloud=CLOUD] FLOATING_IP_ID
    network reserve fixed [ip] [--cloud=CLOUD] FIXED_IP
    network unreserve fixed [ip] [--cloud=CLOUD] FIXED_IP
    network associate floating [ip] [--cloud=CLOUD] [--group=GROUP] [--instance=INS_ID_OR_NAME] [FLOA
    network disassociate floating [ip] [--cloud=CLOUD] [--group=GROUP] [--instance=INS_ID_OR_NAME] [F
```

```
    network create floating [ip] [--cloud=CLOUD] [--pool=FLOATING_IP_POOL]
    network delete floating [ip] [--cloud=CLOUD] FLOATING_IP...
    network list floating pool [--cloud=CLOUD]
    network list floating [ip] [--cloud=CLOUD] [--instance=INS_ID_OR_NAME] [IP_OR_ID]
    network create cluster --group=demo_group
    network -h | --help

Options:
    -h                          help message
    --cloud=CLOUD               Name of the IaaS cloud e.g. india_openstack_grizzly.
    --group=GROUP               Name of the group in Cloudmesh
    --pool=FLOATING_IP_POOL     Name of Floating IP Pool
    --instance=INS_ID_OR_NAME   ID or Name of the vm instance

Arguments:
    IP_OR_ID        IP Address or ID of IP Address
    FIXED_IP        Fixed IP Address, e.g. 10.1.5.2
    FLOATING_IP     Floating IP Address, e.g. 192.1.66.8
    FLOATING_IP_ID  ID associated with Floating IP, e.g. 185c5195-e824-4e7b-8581-703abec4bc01

Examples:
    network get fixed ip --cloud=india 10.1.2.5
    network get fixed --cloud=india 10.1.2.5
    network get floating ip --cloud=india 185c5195-e824-4e7b-8581-703abec4bc01
    network get floating --cloud=india 185c5195-e824-4e7b-8581-703abec4bc01
    network reserve fixed ip --cloud=india 10.1.2.5
    network reserve fixed --cloud=india 10.1.2.5
    network unreserve fixed ip --cloud=india 10.1.2.5
    network unreserve fixed --cloud=india 10.1.2.5
    network associate floating ip --cloud=india --instance=albert-001 192.1.66.8
    network associate floating --cloud=india --instance=albert-001
    network associate floating --cloud=india --group=albert_group
    network disassociate floating ip --cloud=india --instance=albert-001 192.1.66.8
    network disassociate floating --cloud=india --instance=albert-001 192.1.66.8
    network create floating ip --cloud=india --pool=albert-f01
    network create floating --cloud=india --pool=albert-f01
    network delete floating ip --cloud=india 192.1.66.8 192.1.66.9
    network delete floating --cloud=india 192.1.66.8 192.1.66.9
    network list floating ip --cloud=india
    network list floating --cloud=india
    network list floating --cloud=india 192.1.66.8
    network list floating --cloud=india --instance=323c5195-7yy34-4e7b-8581-703abec4b
    network list floating pool --cloud=india
    network create cluster --group=demo_group
```

8.29 nova

Command - nova:

```
Usage:
    nova set CLOUD
    nova info [CLOUD] [--password]
    nova help
    nova [--group=GROUP] ARGUMENTS...

A simple wrapper for the openstack nova command
```

```
Arguments:
    GROUP           The group to add vms to
    ARGUMENTS       The arguments passed to nova
    help            Prints the nova manual
    set             reads the information from the current cloud
                    and updates the environment variables if
                    the cloud is an openstack cloud
    info            the environment values for OS

Options:
    --group=GROUP   Add VM to GROUP group
    --password      Prints the password
    -v              verbose mode
```

8.30 open

Command - open:

```
Usage:
        open FILENAME

ARGUMENTS:
    FILENAME  the file to open in the cwd if . is
              specified. If file in in cwd
              you must specify it with ./FILENAME

Opens the given URL in a browser window.
```

8.31 pause

Command - pause:

```
Usage:
    pause [MESSAGE]

Displays the specified text then waits for the user to press RETURN.

Arguments:
   MESSAGE  message to be displayed
```

8.32 portal

Command - portal:

```
Usage:
    portal start
    portal stop

Examples:
    portal start
        starts the portal and opens the default web page
```

```
    portal stop
        stops the portal
```

8.33 py

Command - py:

```
Usage:
    py
    py COMMAND

Arguments:
    COMMAND   the command to be executed

Description:

    The command without a parameter will be executed and the
    interactive python mode is entered. The python mode can be
    ended with ``Ctrl-D`` (Unix) / ``Ctrl-Z`` (Windows),
    ``quit()``,''exit()``. Non-python commands can be issued with
    ``cmd("your command")``.  If the python code is located in an
    external file it can be run with ``run("filename.py")``.

    In case a COMMAND is provided it will be executed and the
    python interpreter will return to the command shell.

    This code is copied from Cmd2.
```

8.34 q

Command - q:

```
Usage:
    quit

Description:
    Action to be performed whne quit is typed
```

8.35 quit

Command - quit:

```
Usage:
    quit

Description:
    Action to be performed whne quit is typed
```

8.36 quota

Command - quota:

```
Usage:
    quota list [--cloud=CLOUD] [--tenant=TENANT] [--format=FORMAT]

    Prints quota limit on a current project/tenant

Options:
   --format=FORMAT  the output format [default: table]
   --cloud=CLOUD    the cloud name
   --tenant=TENANT  the tenant id

Examples:
    cm quota list
    cm quota list --cloud=india --format=csv
```

8.37 refresh

Command - refresh:

```
Usage:
    refresh on
    refresh off
    refresh list

    switches on and off the refresh for clouds
```

8.38 register

Command - register:

```
Usage:
    register info
    register new
    register clean [--force]
    register list ssh [--format=FORMAT]
    register list [--yaml=FILENAME][--info][--format=FORMAT]
    register cat [--yaml=FILENAME]
    register edit [--yaml=FILENAME]
    register export HOST [--password] [--format=FORMAT]
    register source HOST
    register merge FILEPATH
    register form [--yaml=FILENAME]
    register check [--yaml=FILENAME]
    register test [--yaml=FILENAME]
    register json HOST
    register remote [CLOUD] [--force]
    register india [--force]
    register CLOUD CERT [--force]
    register CLOUD --dir=DIR
    register env [--provider=PROVIDER]
```

```
managing the registered clouds in the cloudmesh.yaml file.
It looks for it in the current directory, and than in
~/.cloudmesh.  If the file with the cloudmesh.yaml name is
there it will use it.  If neither location has one a new
file will be created in ~/.cloudmesh/cloudmesh.yaml. Some
defaults will be provided.  However you will still need to
fill it out with valid entries.

Arguments:

  HOST   the host name
  USER   the user name
  FILEPATH the path of the file
  CLOUD the cloud name
  CERT the path of the certificate
  PROVIDER the provider or type of cloud [Default: openstack]

Options:

  --provider=PROVIDER     Provider to be used for cloud. Values are:
                          openstack, azure, ec2.
  --version=VERSION       Version of the openstack cloud.
  --openrc=OPENRC         The location of the openrc file
  --password              Prints the password
  --force                 ignore interactive questions and execute
                          the action

Description:

    register info
        It looks out for the cloudmesh.yaml file in the current
        directory, and then in ~/.cloudmesh

    register list [--yaml=FILENAME] [--name] [--info]
        lists the clouds specified in the cloudmesh.yaml file. If
        info is specified it also prints the location of the yaml
        file.

    register list ssh
        lists hosts from ~/.ssh/config

    register cat [--yaml=FILENAME]
        outputs the cloudmesh.yaml file

    register edit [--yaml=FILENAME]
        edits the cloudmesh.yaml file

    register export HOST [--format=FORMAT]

          prints the contents of an openrc.sh file based on the
          information found in the cloudmesh.yaml file.

    register remote CLOUD [--force]

          reads the Openstack OPENRC file from a remote host that
          is described in cloudmesh.yaml file. We assume that
          the file has already a template for this host. If
          not it can be created from other examples before
```

```
          you run this command.

          It uses the OS_OPENRC variable to locate the file and
          copy it onto your computer.

    register merge FILENAME
        Replaces the TBD in cloudmesh.yaml with the contents
        present in the named file

    register form [--yaml=FILENAME]
        interactively fills out the form wherever we find TBD.

    register check [--yaml=FILENAME]
        checks the yaml file for completness

    register test [--yaml=FILENAME]
        checks the yaml file and executes tests to check if
        we can use the cloud. TODO: maybe this should be in
        a test command

    register json host
        displays the host details in json format

    register remote CLOUD
        registers a remote cloud and copies the openrc file
        specified in the credentials of the cloudmesh.yaml

    register CLOUD CERT [--force]
        Copies the CERT to the ~/.cloudmesh/clouds/host directory
        and registers that cert in the coudmesh.yaml file.


    register CLOUD --dir
        Copies the entire directory from the cloud and puts it in
        ~/.cloudmesh/clouds/host
        For india, The directory would be copied to
        ~/.cloudmesh/clouds/india

    register env [--provider=PROVIDER] [HOSTNAME]
        Reads env OS_* variables and registers a new cloud in yaml,
        interactively. Default PROVIDER is openstack and HOSTNAME
        is localhost.
```

8.39 reservation

Command - reservation:

```
Usage:
    reservation info --user=USER --project=PROJECT
    reservation list [--name=NAME]
                     [--user=USER]
                     [--project=PROJECT]
                     [--hosts=HOSTS]
                     [--start=TIME_START]
                     [--end=TIME_END]
                     [--format=FORMAT]
```

```
    reservation delete [all]
                       [--user=USER]
                       [--project=PROJECT]
                       [--name=NAME]
                       [--start=TIME_START]
                       [--end=TIME_END]
                       [--hosts=HOSTS]
    reservation delete --file=FILE
    reservation update --name=NAME
                      [--start=TIME_START]
                       [--end=TIME_END]
                       [--user=USER]
                       [--project=PROJECT]
                       [--hosts=HOSTS]
                       [--description=DESCRIPTION]
    reservation add --name=NAME
                    [--start=TIME_START]
                    [--end=TIME_END]
                    [--user=USER]
                    [--project=PROJECT]
                    [--hosts=HOSTS]
                    [--description=DESCRIPTION]
    reservation add --file=FILE

Arguments:

    NAME            Name of the reservation
    USER            Registration will be done for this user
    PROJECT         Project to be used
    HOSTS           Hosts to reserve
    TIME_START      Start time of reservation
    TIME_END        End time of reservation
    FORMAT          Format of output
    DESCRIPTION     Description for reservation
    FILE            File that contains reservation data to be added/ deleted

Options:

    --name=NAME           Names of the reservation
    --user=USER           user name
    --project=PROJECT     project id
    --start=TIME_START    Start time of the reservation, in
                          MM/DD/YYYY at hh:mm aa format. (default value: 01/01/1901 at 12:00 am])
    --end=TIME_END        End time of the reservation, in
                          MM/DD/YYYY at hh:mm aa format. (default value: 12/31/2100 at 11:59 pm])
    --host=HOSTS          host name
    --description=DESCRIPTION  description summary of the reservation
    --file=FILE           Adding multiple reservations from one file
    --format=FORMAT       Format is either table, json, yaml or csv
                          [default: table]

Description:

    reservation info
        lists the resources that support reservation for
        a given user or project.
```

8.40 reset

Command - reset:

```
  Usage:
      reset

Description:

    DANGER: This method erases the database.


Examples:
    clean
```

8.41 rsync

Command - rsync:

```
Usage:
    rsync ARGUMENTS...

A simple wrapper for rsync command

Arguments:
    ARGUMENTS       The arguments passed to nova

Options:
    -v              verbose mode
```

8.42 secgroup

Command - secgroup:

```
Usage:
    secgroup list [--cloud=CLOUD] [--tenant=TENANT]
    secgroup create [--cloud=CLOUD] [--tenant=TENANT] LABEL
    secgroup delete [--cloud=CLOUD] [--tenant=TENANT] LABEL
    secgroup rules-list [--cloud=CLOUD] [--tenant=TENANT] LABEL
    secgroup rules-add [--cloud=CLOUD] [--tenant=TENANT] LABEL FROMPORT TOPORT PROTOCOL CIDR
    secgroup rules-delete [--cloud=CLOUD] [--tenant=TENANT] LABEL FROMPORT TOPORT PROTOCOL CIDR
    secgroup refresh [--cloud=CLOUD]
    secgroup -h | --help
    secgroup --version

Options:
    -h                  help message
    --cloud=CLOUD       Name of the IaaS cloud e.g. india_openstack_grizzly.
    --tenant=TENANT     Name of the tenant, e.g. fg82.

Arguments:
    LABEL         The label/name of the security group
    FROMPORT      Staring port of the rule, e.g. 22
    TOPORT        Ending port of the rule, e.g. 22
```

```
    PROTOCOL      Protocol applied, e.g. TCP,UDP,ICMP
    CIDR          IP address range in CIDR format, e.g., 129.79.0.0/16

Description:
    security_group command provides list/add/delete
    security_groups for a tenant of a cloud, as well as
    list/add/delete of rules for a security group from a
    specified cloud and tenant.


Examples:
    secgroup list --cloud india --tenant fg82
    secgroup rules-list --cloud india --tenant fg82 default
    secgroup create --cloud india --tenant fg82 webservice
    secgroup rules-add --cloud india --tenant fg82 webservice 8080 8088 TCP "129.79.0.0/16"
```

8.43 select

Command - select:

```
Usage:
    select image [CLOUD] [--refresh]
    select flavor [CLOUD] [--refresh]
    select cloud [CLOUD]
    select key [CLOUD]

selects interactively the default values

Arguments:

  CLOUD    the name of the cloud

Options:

  --refresh   refreshes the data before displaying it
              from the cloud
```

8.44 server

Command - server:

```
Usage:
    server

Options:
  -h --help
  -v       verbose mode

Description:
  Starts up a REST service and a WEB GUI so one can browse the data in an
  existing cloudmesh database.

  The location of the database is supposed to be in
```

```
    ~/.cloud,esh/cloudmesh.db
```

8.45 shell

Command - shell:

```
Usage:
   shell ARGUMENTS...

Description:
    Executes a shell command
```

8.46 ssh

Command - ssh:

```
Usage:
    ssh table
    ssh list [--format=FORMAT]
    ssh cat
    ssh register NAME PARAMETERS
    ssh ARGUMENTS


conducts a ssh login on a machine while using a set of
registered machines specified in ~/.ssh/config

Arguments:

  NAME        Name or ip of the machine to log in
  list        Lists the machines that are registered and
              the commands to login to them
  PARAMETERS  Register te resource and add the given
              parameters to the ssh config file.  if the
              resoource exists, it will be overwritten. The
              information will be written in /.ssh/config

Options:

   -v       verbose mode
   --format=FORMAT   the format in which this list is given
                     formats incluse table, json, yaml, dict
                     [default: table]

   --user=USER       overwrites the username that is
                     specified in ~/.ssh/config

   --key=KEY         The keyname as defined in the key list
                     or a location that contains a pblic key

Description:

    ssh list
```

```
        lists the hostsnames  that are present in the
        ~/.ssh/config file

    ssh cat
        prints the ~/.ssh/config file

    ssh table
        prints contents of the ~/.ssh/config file in table format

    ssh register NAME PARAMETERS
        registers a host i ~/.ssh/config file
        Parameters are attribute=value pairs
        Note: Note yet implemented

    ssh ARGUMENTS
        executes the ssh command with the given arguments
        Example:
            ssh myhost

                conducts an ssh login to myhost if it is defined in
                ~/.ssh/config file
```

8.47 submit

Command - submit:

```
Usage:
    submit ARGUMENTS...

We do not yet know what this command will do ;-)

Arguments:
    ARGUMENTS       The arguments passed to nova

Options:
    -v              verbose mode
```

8.48 sync

Command - sync:

```
Usage:
    sync put [--cloud=CLOUD] LOCALDIR [REMOTEDIR]
    sync get [--cloud=CLOUD] REMOTEDIR LOCALDIR

A simple wrapper for the openstack nova command

Arguments:
    LOCALDIR        A directory on local machine
    REMOTEDIR       A directory on remote machine

Options:
    --cloud=CLOUD   Sync with cloud
```

8.49 usage

Command - usage:

```
Usage:
    usage list [--cloud=CLOUD] [--start=START] [--end=END] [--tenant=TENANT] [--format=FORMAT]

    Show usage data.

Options:
   --format=FORMAT  the output format [default: table]
   --cloud=CLOUD    the cloud name
   --tenant=TENANT  the tenant name
   --start=START    Usage range start date ex 2012-01-20, default is: 4 weeks ago
   --end=END        Usage range end date, ex 2012-01-20, default is: tomorrow


Examples:
    cm usage list
```

8.50 var

Command - var:

```
Usage:
    var list
    var delete NAMES
    var NAME=VALUE
    var NAME
Arguments:
    NAME    Name of the variable
    NAMES   Names of the variable separated by spaces
    VALUE   VALUE to be assigned
special vars date and time are defined
```

8.51 verbose

Command - verbose:

```
Usage:
    verbose (True | False)
    verbose

NOTE: NOT YET IMPLEMENTED.
If it sets to True, a command will be printed before execution.
In the interactive mode, you may want to set it to False.
When you use scripts, we recommend to set it to True.

The default is set to False

If verbose is specified without parameter the flag is
toggled.
```

8.52 version

Command - version:

```
Usage:
   version [--format=FORMAT] [--check=CHECK]

Options:
    --format=FORMAT  the format to print the versions in [default: table]
    --check=CHECK    boolean tp conduct an additional check [default: True]

Description:
    Prints out the version number
```

8.53 vm

Command - vm:

```
Usage:
    vm default [--cloud=CLOUD][--format=FORMAT]
    vm refresh [--cloud=CLOUD]
    vm boot [--name=NAME]
            [--cloud=CLOUD]
            [--image=IMAGE_OR_ID]
            [--flavor=FLAVOR_OR_ID]
            [--group=GROUP]
            [--secgroup=SECGROUP]
            [--key=KEY]
            [--dryrun]
    vm start [NAME]...
             [--group=GROUP]
             [--cloud=CLOUD]
             [--force]
    vm stop [NAME]...
            [--group=GROUP]
            [--cloud=CLOUD]
            [--force]
    vm delete [NAME]...
              [--group=GROUP]
              [--cloud=CLOUD]
              [--force]
    vm ip assign [NAME]...
              [--cloud=CLOUD]
    vm ip show [NAME]...
               [--group=GROUP]
               [--cloud=CLOUD]
               [--format=FORMAT]
               [--refresh]
    vm login [NAME] [--user=USER]
             [--ip=IP]
             [--cloud=CLOUD]
             [--key=KEY]
             [--command=COMMAND]
    vm list [NAME_OR_ID]
            [--cloud=CLOUD|--all]
            [--group=GROUP]
```

```
            [--format=FORMAT]
            [--refresh]
    vm status [--cloud=CLOUD]
    vm info [--cloud=CLOUD]
            [--format=FORMAT]

Arguments:
    COMMAND        positional arguments, the commands you want to
                   execute on the server(e.g. ls -a) separated by ';',
                   you will get a return of executing result instead of login to
                   the server, note that type in -- is suggested before
                   you input the commands
    NAME           server name. By default it is set to the name of last vm from database.
    NAME_OR_ID     server name or ID
    KEYPAIR_NAME   Name of the openstack keypair to be used to create VM. Note this is not a path to

Options:
    --ip=IP          give the public ip of the server
    --cloud=CLOUD    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
    --detail         for table print format, a brief version
                     is used as default, use this flag to print
                     detailed table
    --flavor=FLAVOR_OR_ID  give the name or id of the flavor
    --group=GROUP          give the group name of server
    --secgroup=SECGROUP    security group name for the server
    --image=IMAGE_OR_ID    give the name or id of the image
    --key=KEY        specify a key to use, input a string which
                     is the full path to the private key file
    --keypair_name=KEYPAIR_NAME   Name of the openstack keypair to be used to create VM.
                                  Note this is not a path to key.
    --user=USER      give the user name of the server that you want
                     to use to login
    --name=NAME      give the name of the virtual machine
    --force          delete vms without user's confirmation
    --command=COMMAND
                     specify the commands to be executed



Description:
    commands used to boot, start or delete servers of a cloud

    vm default [options...]     Displays default parameters that are set for VM boot.
    vm boot [options...]        Boots servers on a cloud, user may specify
                                flavor, image .etc, otherwise default values
                                will be used, see how to set default values
                                of a cloud: cloud help
    vm start [options...]       Starts a suspended or stopped vm instance.
    vm stop [options...]        Stops a vm instance .
    vm delete [options...]      delete servers of a cloud, user may delete
                                a server by its name or id, delete servers
                                of a group or servers of a cloud, give prefix
                                and/or range to find servers by their names.
                                Or user may specify more options to narrow
                                the search
    vm floating_ip_assign [options...]   assign a public ip to a VM of a cloud
```

```
    vm ip_show [options...]     show the ips of VMs
    vm login [options...]       login to a server or execute commands on it
    vm list [options...]        same as command "list vm", please refer to it
    vm status [options...]      Retrieves status of last VM booted on cloud and displays it.

Tip:
    give the VM name, but in a hostlist style, which is very
    convenient when you need a range of VMs e.g. sample[1-3]
    => ['sample1', 'sample2', 'sample3']
    sample[1-3,18] => ['sample1', 'sample2', 'sample3', 'sample18']
```

CHAPTER 9

Exercises

9.1 Assignment A: Prerequisite

- A.1) Get account on futuresystems.org or any other cloud you have access to. In case you take a class that uses cloudmesh and futuresystems, make sure to be in a valid project. Communicate with your teacher who will let you know.
- A.2) Why do you need to start assignment A.1 today and can not wait with it till the day before the due date?

9.2 Assignment B: IaaS

- A.1) Is prerequisite
- B.1) Install cloudmesh on local machine (we recommend a virtual box)
- B.2) Start and stop vms on the kilo cloud
- B.3) Why do i need to shut down my VM?
- B.4) Can I leave my VM simply running?
- B.5) What will happen to your VM when there is a power outage that shuts down the cloud?
- B.6) Assume you create 2 VMs. How do you log in securely from one to the other VM. What needs to be done?

9.3 Assignment C: Ansible

- A.1) Is prerequisite
- C.1) Install cloudmesh on local machine (we recommend a virtual box)
- C.2) Develop automated script for the installation
- C.3) Generate an image on kilo cloud that uses the automated script and install s cloudmesh in the image
- C.4) Develop an ansible script that generates an image that has cloudmesh installed in it
- C.5) Bonus: use docopt to select from a command that you develop which OS is used and conduct the ansible install for the OS

that you chose.

9.4 Assignment D: Key Management

- D.1) What is an RSA key?
- D.2) Where are such keys stored in a user environment?
- D.3) Describe the procedures needed to use the default key (rsa) in Openstack with the openstack client commands.
- D.4) Describe the procedures to use the default key (rsa) in cloudmesh client
- D.5) do B.6 How can this be generalized to n virtual machines. Can you write a script?

CHAPTER 10

API

10.1 Cloud Database

Cloudmesh contains a convenient Cloud database to store its objects. It also contains simple functions to synchronize the database with objects that are found in clouds. This includes images, flavors, and virtual machines.

The clouds are defined in ~/.cloudmesh/cloudmesh.yaml

All you need to do is to create a cm object:

```
cm = CloudmeshDatabase(cm_user="gregor")
```

To update a specific set of cloud object such as the flavors on the cloud india you simply can say:

```
cm.update("flavor", "india")
```

Other examples include

cm.update(“image”, “india”) cm.update(“vm”, “india”)

Multiple updates and clouds can be introduced with a parametrized call:

```
cm.update("vm,flavor,image", "india,aws,azure")
```

In our example all clouds specified update the virtual machines, images, and flavors in the database. Please note that the keywords used are singular.

Once the data is in the database its easy to query it either with the native query functions or with specialized find functions exposed to the cm object.

To query for example a vm with the name “gregor-001” you can use

vm = cm.find(“vm”, name=”gregor-001”).first()

Using the method:

```
d = cm.o_to_d(vm)
```

returns a dict in the object d. Alternatively you could also use the native database format and for example get information via:

```
vm.name
vm.status
....
```

In some cases using dicts is more convenient. You may want to chose if you use the native form or the dict representation.

10.2 Updating an element in the database

The cloud related data have a number of attributes that make it easy to identify them. The most important one is 'cm_id' which presents in human readable format a unique id for the object in the database.

The id is generated fir the *getID* method.

Let us assume the following setup for our example:

```
cm = CloudmeshDatabase(cm_user="gregor")
```

this will create a cm database object in which the user *gregor* stores its values. First we need to get a dictionary that we may want to store and modify in the database. We can obtain such an object changeme from a live cloud with:

```
cloud = OpenStack_libcloud(
    "acloudnamedefinedin_cludmesh.yaml",
    cm_user="yourusernameonthecloud")
flavors = cloud.list("flavor", output="flat")

internal_id = "1"

changeme = flavors[internal_id]
```

However such an object could also be created by hand. To store the element in the database we first need to generate a unique cm_id. In our case we use the cloud object type (here flavor), the unique internal id that we obtain for each flavor, and the name of the cloud on which the object belongs to.:

```
changeme["cm_id"] = cm.getID("flavor", internal_id, "india")
```

Just to be sure lest set the type to flavor:

```
changeme["cm_type"] = "flavor"
```

Now let us change the label of the object to:

```
changeme["label"] = "newlabel"
```

To update the new object to the database use:

```
cm.update_from_dict(f)
cm.save()
```

CHAPTER 11

Hacking

11.1 Contributing

The project is open source and if you like you can help. Here are some contribution guidelines:

- we use the Apache 2.0 licence
- we recommend to use pycharm for editing and utlize the *Inspect Code* feature regularly on the files you modify and fix the sensible pep8 warnings. Look at other warnings and errors
- use `.format` instead of `%` in print statements
- use `from __future__ import print_function` at the beginning of the file and use `print("msg")` instead of `print msg`
- use as much python 3 like code but use python 2
- use `class name(object):` and not just `class name:`
- use `Console.error` when printing errors if you develop a command for the commandline
- use `Console.ok` when printing something that confirms the action is ok. In many cases you may not want o use print, but Console.ok. Please note that Console.ok, takesa string argument.
- use `ConfigDict` when reading yaml files. It even allows you to read a yaml file in order, make changes to it and write it back
- use nosetests for testing your programs

  An example is provided in `tests/test_sample.py`:

```
nosetests -v --nocapture tests/test_sample.py
```

  you can copy this to `tests/test_yourfile.py` and then replace in that file the occurrence of `_sample` with `_yourfile`

  make sure to create meaningful tests.
- the documentation is write in RST http://sphinx-doc.org/rest.html
- look at http://python-future.org/compatible_idioms.html for tips to make python 2 look more like python 3. Be careful with dicts to make them not inefficient.
- we are using https://pypi.python.org/pypi/gitchangelog for creating changelogs automatically. Thus you need to use a prefix in any commit. It includes the type and the user for which the commit is relevant. Examples are:

```
    chg: usr:    simple changes relevant for users (spelling, ...)
    fix: usr:    major changes for users
    new: usr:    new feature for users

    chg: dev:    simple changes relevant for developers (spelling, ...)
    fix: dev:    major changes for developers
    new: dev:    new feature for developers
```

11.2 Editor

Use PyCharam.

We made bad experience with people using editors other than emacs, vi, and PyCharm. When working on Windows make sure your editor handles newlines properly with git.

11.3 Documentation

Creating the documentation with sphinx is easy

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

View the documentation

```
$ make view
```

11.4 Git

11.4.1 Closing Issues via Commit Messages

To close an issue on github issues, you can use it in your commit messages as follows

git commit -m "Fix problem xyz, fixes #12"

11.4.2 SSH keys

You can get a list of public ssh keys in plain text format by visiting:

https://github.com/{user}.keys

11.4.3 Sheetsheet

- https://github.com/tiimgreen/github-cheat-sheet
- https://training.github.com/kit/downloads/github-git-cheat-sheet.pdf
- http://byte.kde.org/~zrusin/git/git-cheat-sheet.svg
- http://www.emoji-cheat-sheet.com

11.4.4 Empty Commits

Commits can be pushed with no code changes by adding –allow-empty:

```
$ git commit -m "Big-ass commit" --allow-empty
```

11.4.5 Styled Git Log

```
$ git log --all --graph --pretty=format:'%Cred%h%Creset -%C(auto)%d%Creset %s %Cgreen(%cr) %C(bold bl
```

11.4.6 Tags

Tags are only created by Gregor von Laszewski.

Create a tag. Always use x.y.z

```
$ make tag
```

Remove a tag

```
$ make rmtag
```

11.4.7 Publish on Github

The documentation is only pushed by Gregor von Laszewski.

```
$ make publish
```

11.4.8 Logging

```
from cloudmesh_client.common.LogUtil import LogUtil

log = LogUtil.get_logger()
log.info("Cloud: " + cloud + ", Arguments: " + str(arguments))
```

CHAPTER 12

Appendix

12.1 Developer Notes

12.1.1 OpenStack API Examples

The following documents show how to access Openstack with the native Openstack API.

- http://docs.pistoncloud.com/getting_started/tutorials/api_tutorial.html
- http://docs.openstack.org/developer/python-novaclient/api.html
- https://albertomolina.wordpress.com/2013/11/20/how-to-launch-an-instance-on-openstack-iii-python-novaclient-library/
- http://blog.briancurtin.com/posts/nice-apis-limits-in-openstack-sdk.html
- https://community.hpcloud.com/article/retrieving-your-resource-limits-quotas

12.2 Changes

12.2.1 Summary of Changes

TBD

12.2.2 Changelog

%%version%% (unreleased)

- Comet: more cleaning up of code and debug messages. [fugangwang]
- Comet: cleaning up code/docs based on currently implemented features. [fugangwang]
- Updating comet documentation. [fugangwang]
- Comet: enabling powering off/reboot/reset all nodes for an active computeset. [fugangwang]
- Comet: changing docopt. [fugangwang]
- Comet: adding notes about powering off computeset. [fugangwang]
- Comet: updating quickstart guide. [fugangwang]

- Comet: Changing power management of nodes; Adding support of allocation and customized walltime. [fugangwang]
- Chg;dev: pip instalation is not working notice. [Gregor von Laszewski]

2.0.4 (2016-02-11)

- 2.0.3. [Gregor von Laszewski]
- Version 2.0.3. [Gregor von Laszewski]

2.0.3 (2016-02-10)

- 2.0.2. [Gregor von Laszewski]
- Version 2.0.2. [Gregor von Laszewski]

2.0.2 (2016-02-10)

- 2.0.1. [Gregor von Laszewski]
- Version 2.0.1. [Gregor von Laszewski]

2.0.1 (2016-02-10)

- 2.0.0. [Gregor von Laszewski]
- Version 2.0.0. [Gregor von Laszewski]

2.0.0 (2016-02-10)

- 1.1.6. [Gregor von Laszewski]
- Version 1.1.6. [Gregor von Laszewski]

1.1.6 (2016-02-09)

New

- Adding a "cm key load" command that loads the keys from the yaml file. [Gregor von Laszewski]
- Auto-refresh images and flavors if the database does not contain values for it. [Gregor von Laszewski]
- Reading default values from the db or the yaml file at cm start. [Gregor von Laszewski]
- Adding a homework about keys. [Gregor von Laszewski]
- Pointer to the documentation on RTD. [Gregor von Laszewski]

Changes

- Improved the "cm register info" command. [Gregor von Laszewski]
- New location of the Cloudmesh presentation. [Gregor von Laszewski]

Other

- Version 1.1.5. [Gregor von Laszewski]

1.1.5 (2016-02-02)

- Version 1.1.4. [Gregor von Laszewski]

1.1.4 (2016-02-01)

New

- Change link to presentation. [Gregor von Laszewski]
- Adding the ppt. [Gregor von Laszewski]
- Improved script examples. [Gregor von Laszewski]
- Introduce variables to make .cloudmesh dependent scripts. [Gregor von Laszewski]
- Commenting the cm scripts for the manual. [Gregor von Laszewski]
- Added a simplw quick start guide. [Gregor von Laszewski]
- Adding cybera configuration instructions. [Gregor von Laszewski]
- Removing IU juno cloud from the yaml file as it will be shut of Jan 30, 2016. [Gregor von Laszewski]
- One line curl based install for centos. [Gregor von Laszewski]
- Using bash prompt in sphinx documentation. [Gregor von Laszewski]

Changes

- Dev spelling and generalizing register remote. [Gregor von Laszewski]
- Add chameleon auth url. [Gregor von Laszewski]
- Removing juno cloud from most of the code. [Gregor von Laszewski]

Fix

- Simplified install instructions. [Gregor von Laszewski]

Other

- Adding comet CLI quick start guide. [fugangwang]
- Console URL could be unavailable. [fugangwang]
- Computeset state changed from started to running after slurm integration. [fugangwang]
- Version 1.1.3. [Gregor von Laszewski]

1.1.3 (2016-01-26)

New

- Start of a reference card and a quick start guide. [Gregor von Laszewski]
- Start of an integrated manual. [Gregor von Laszewski]
- Makeing the api documentation work. [Gregor von Laszewski]
- Added more user manual pages. [Gregor von Laszewski]

Changes

- New: homework examples. [Gregor von Laszewski]
- Improving the documentation. [Gregor von Laszewski]
- Add template for ec2 chameleon cloud. [Gregor von Laszewski]
- New chameleon cloud EC2 setup instructions. [Gregor von Laszewski]
- Improve the configuration documentation. [Gregor von Laszewski]
- Added –cloud=general note when no general default group is set. [Gregor von Laszewski]

Fix

- Update ubuntu and centos instructions. [Gregor von Laszewski]
- Improced OSX install instructions. [Gregor von Laszewski]
- Enable date printing in hpc command submission. [Gregor von Laszewski]
- Removed docopt parsing from cm main command to allow parsing of parameters in cm command. [Gregor von Laszewski]

Other

- Usr: new: improve API documentation. [Gregor von Laszewski]
- Apidoc dir. [Gregor von Laszewski]
- Version 1.1.2. [Gregor von Laszewski]

1.1.2 (2016-01-13)

- New state defined on comet backend. [fugangwang]
- Version 1.1.1. [Gregor von Laszewski]

1.1.1 (2016-01-13)

New

- Force cm help at install time to create the yaml files. [Gregor von Laszewski]

Other

- Version 1.1.0. [Gregor von Laszewski]

1.1.0 (2016-01-13)

New

- Making a dickefile that creates a cloudmesh image. [Gregor von Laszewski]
- Adding an automatic refresh command for clouds. [Gregor von Laszewski]
- Add id or name management to booting images. [Gregor von Laszewski]
- Add uuid to image list. [Gregor von Laszewski]
- Add template for cybera cloud. [Gregor von Laszewski]
- Improve key command options. [Gregor von Laszewski]
- Simplified the table when just saying "default" [Gregor von Laszewski]
- Change the way cm is started. [Gregor von Laszewski]
- Added cluster command prototype. [Gregor von Laszewski]
- Template for a cluster command to be implemented similar to the old cloudmesh. [Gregor von Laszewski]
- Added yaml file variables to the manual. [Gregor von Laszewski]
- Allow "history" without the list option to behave just like history list. [Gregor von Laszewski]
- Add variable support read from cloudmesh.yaml file; fix echo command. [Gregor von Laszewski]
- Adding a bebug toggle command. [Gregor von Laszewski]
- Added docker makefile. [Gregor von Laszewski]
- Added a check command template. [Gregor von Laszewski]
- Adding more commands to the shell topic. [Gregor von Laszewski]
- Add simple history command. [Gregor von Laszewski]
- Adding var command and reorganizing documentation. [Gregor von Laszewski]
- Add comment example. [Gregor von Laszewski]
- Detect if the line is a existing file than execute it. [Gregor von Laszewski]
- Add a shell command so we can execute them from cm. [Gregor von Laszewski]
- Added user manual for hpc kill. [Erika Dsouza]
- Documentation for hpc experiment runs. [Gregor von Laszewski]
- Moved the hpc experiment commands to hpc run commands. [Gregor von Laszewski]
- Easy way to list remote experiments. [Gregor von Laszewski]
- List of experiments on remote clusters. [Gregor von Laszewski]
- Batch provider template. [Gregor von Laszewski]
- Database model for scripts. [Gregor von Laszewski]
- Transformed the hpc.run to a proper dict return. [Gregor von Laszewski]

- Add credentials for project id to yaml file. [Gregor von Laszewski]
- Start of comet command definition. [Gregor von Laszewski]
- Added hpc info and queue user manual. [Erika Dsouza]

Changes

- Def: merge. [Gregor von Laszewski]
- Testing key management. [Gregor von Laszewski]
- Do not overwrite existing default values from cloudmesh.yaml in the database. [Gregor von Laszewski]
- Fix default table printer, add -v to flavor. [Gregor von Laszewski]
- Moving towards kilo as default cloud. [Gregor von Laszewski]
- Addding demo script. [Erika Dsouza]
- Adding user manual for hpc run when transfering script. [Erika Dsouza]
- Introducing a better separation of the provider fro clouds. [Gregor von Laszewski]
- Restructure the HPC provider in a separate subdir and introducing a factory. [Gregor von Laszewski]
- Fix the sync command and also the topic assignment, start of an echocommand. [Gregor von Laszewski]
- Sr: improved indentation. [Gregor von Laszewski]
- Improved heading. [Gregor von Laszewski]

Fix

- Fix the image nose test. [Gregor von Laszewski]
- Update the coloms in the image command. [Gregor von Laszewski]
- Added commands in cm shell. [Erika Dsouza]

Other

- Check and enabling secgroup rule to allow ssh login. [fugangwang]
- Set secgroup rule or default to allow ssh login. [fugangwang]
- Specifying default sshkey to login VM. [fugangwang]
- Always check if there is any allocated IPs already before acquiring new ones. [fugangwang]
- Walltime_mins now required when booting a group of nodes due to the integration with slurm. [fugangwang]
- Removing debugging message of nics management. [fugangwang]
- More robust handling of netid parameter. [fugangwang]
- Nics parameter for boot vm in india kilo. [fugangwang]
- Controlling if traceback is printed out when exception occured. [fugangwang]
- Adding missing keystone client requirement. [Gregor von Laszewski]
- Usr: chg: adding hpc demo script for group. [Erika Dsouza]

- Separating action and display so the power functions could be properly called by the portal. [fugangwang]
- Usr: chg: adding virtualenv to osx. [Erika Dsouza]
- Displaying the unescaped url in case of non compatible os. [fugangwang]
- Making computeset_id semantically correct. [fugangwang]
- Console_url for portal embedding. [fugangwang]
- Username missing in config for comet. [fugangwang]
- Minor. [Hyungro Lee]
- Minor. [Hyungro Lee]
- Minor. [Hyungro Lee]
- Update README.rst. [Hyungro Lee]
- Minor. [Hyungro Lee]
- Hello world heat example. [Hyungro Lee]
- Excluding 'completed' computeset as they have no use at all. [fugangwang]
- Handling redirection of the console url internally and open the final url in browser. [fugangwang]
- Fixing platform check. [fugangwang]
- Better error handling when failed to authenticate. [fugangwang]
- Added console command; better/consistent parameter handling. [fugangwang]
- Introduce portal command. [Gregor von Laszewski]
- Changing prefix of comet auth credentials. [fugangwang]
- Version 1.0.2. [Gregor von Laszewski]

1.0.2 (2015-11-25)

New

- Starting to import the heat templates. [Gregor von Laszewski]

Other

- Added httpsig in setup.py so pip install can install the library. [fugangwang]
- Version 1.0.1. [Gregor von Laszewski]

1.0.1 (2015-11-25)

- Pep8 fixes. [Mangirish Wagle]
- Supporting both USERPASS and APIKEY auth method for comet. [fugangwang]
- Choice of auth provider for comet as USERPASS/APIKEY in yaml config. [fugangwang]
- Version 1.0.0. [Gregor von Laszewski]

1.0.0 (2015-11-24)

New

- Chg: add max_width for columns to dict printer. [Gregor von Laszewski]
- Fix to the regiter command upon first registration of a remote cloud. [Gregor von Laszewski]

Changes

- Updated yaml file format. [Gregor von Laszewski]
- Fix the userid vs user fields caused by double nameing in header of slurm. [Gregor von Laszewski]
- Add image for openmpi. [Gregor von Laszewski]
- Dev removing general. [Erika Dsouza]

Other

- Fix for slurm table to distingush "user" from "user ". E.g. replace the space wit _ [Gregor von Laszewski]
- Version 0.7.7. [Gregor von Laszewski]

0.7.7 (2015-11-18)

New

- Creating cloudmesh yaml file on first call if it does not exist. [Gregor von Laszewski]

Changes

- Adding error msg where we need to get the cloudmesh yaml file from when it does not exist. [Gregor von Laszewski]
- Fix the install in setup.py. [Gregor von Laszewski]

Other

- Fixing data_files format as tuple. [fugangwang]
- More error handling for nucleus service error. [fugangwang]
- Displaying mac/ip addresses for computeset. [fugangwang]
- Fixing for mac/ip address. [fugangwang]
- Version 0.7.6. [Gregor von Laszewski]

0.7.6 (2015-11-15)

- Version 0.7.5. [Gregor von Laszewski]

0.7.5 (2015-11-15)

- Version 0.7.4. [Gregor von Laszewski]

0.7.4 (2015-11-15)

New

- Template profiles for additional scripts. [Gregor von Laszewski]
- Implement a prototype (non working) for cm default. It does not yet read from yaml files. THis may have not to be done here, but at startup of cm if defaults are defined and the values in the database are not set. [Gregor von Laszewski]
- Demo on how touse py. [Gregor von Laszewski]
- Adding a scripts directory with sample scripts. [Gregor von Laszewski]
- When a single argument is given to cm that ends in .cm it is interpreted as script. cm demo.cm. [Gregor von Laszewski]
- Add a exec script command so it also works with cm –script=filename. [Gregor von Laszewski]
- A simple demo script. [Gregor von Laszewski]
- Allowing commenst in cmd with #, // and /*, they must be on one line and at the beginning. [Gregor von Laszewski]
- Added manual for flavor command. [ehdsouza]
- Added manual for image. [ehdsouza]
- Adding the rack diagram. [Gregor von Laszewski]
- Adapting the list command to the new comet rest interface. [Gregor von Laszewski]
- Creating a simple comet command. [Gregor von Laszewski]
- Introducing dynamic class loading. [Gregor von Laszewski]
- Whois data structure. [Gregor von Laszewski]

Changes

- Placehoder for demo script. [Gregor von Laszewski]
- Updating the user documentation. [Gregor von Laszewski]
- Fix the default database class. [Gregor von Laszewski]
- Set default cloud and group at startup of cm if they do not exists. [Gregor von Laszewski]
- Change CloudProvider.os_environ to CloudProvider.set. [Gregor von Laszewski]

Fix

- Removed debug message. [ehdsouza]
- Added uniform user message for flavor refresh. [ehdsouza]
- Modifications to image list. [ehdsouza]

- Refactored default list to use cm database. [ehdsouza]

Other

- Changes to support more power actions. [fugangwang]
- Remove debug messages. [Gregor von Laszewski]
- Set a cluster default. [Gregor von Laszewski]
- Dev: usr: improved manual and pep8 formating. [Gregor von Laszewski]
- Removing the user name. [Gregor von Laszewski]
- More friendly output for power off. [fugangwang]
- Comet client and cluster updated due to changes on API. [fugangwang]
- Updating comet based on API and configuration changes. [fugangwang]
- Chk: dev: improving group command. [Gregor von Laszewski]
- Update index.rst. [Gregor von Laszewski]
- Version 0.7.3. [Gregor von Laszewski]

0.7.3 (2015-10-31)

New

- Fix the usage command and refactor the TableParser, chnage tables to Printer. [Gregor von Laszewski]
- Switching to classmethods. [Gregor von Laszewski]
- Add a function cm register new that copies the etc/yaml file into .cloudmesh. It does not ask .... [Gregor von Laszewski]
- Changing the credentials so that they can be used directly in openstack api calls, removing OS_VERSION, as already defined as cm_type_version. [Gregor von Laszewski]

Changes

- Improve the base class. [Gregor von Laszewski]
- Replace the authentication class. [Gregor von Laszewski]

Other

- Version 0.7.2. [Gregor von Laszewski]

0.7.2 (2015-10-30)

New

- Makong the *cm limits* command working. [Gregor von Laszewski]
- Refactorization of Authenticator to CloudProvider, introduction of ListResources. [Gregor von Laszewski]

- Adding a command *cm register source CLOUD* that adds the environment variables to your shell. [Gregor von Laszewski]
- Adding elementary support for chameleon cloud. [Gregor von Laszewski]
- Fix the cm register list command. [Gregor von Laszewski]
- Fixing cm register (list, ssh), adding some format options. [Gregor von Laszewski]
- Replace the "register rc" and and a "register export" [Gregor von Laszewski]
- Introducing inteligent serach on id, name, and uuid in image list. [Gregor von Laszewski]
- Add flavor command. [Gregor von Laszewski]
- Changing some of the cloud api methods to use list and get more uniformly. [Gregor von Laszewski]
- Working on an alternative remote registration. [Gregor von Laszewski]
- Working on an alternative remote registration. [Gregor von Laszewski]
- Introducing a simple table parser. [Gregor von Laszewski]
- Test and documentation for usage. [ehdsouza]
- Makeing the color on off cpommand much more convenient and working. [Gregor von Laszewski]
- Adding format to comet list an ll commands. [Gregor von Laszewski]
- Comet docs command. [Gregor von Laszewski]
- Comet cluster list command. [Gregor von Laszewski]
- Introducing a simple comet command. [Gregor von Laszewski]

Changes

- First fixes to the cm nova command. [Gregor von Laszewski]
- Readding changing of permissions. [Gregor von Laszewski]
- Updating the documentation for *cm register* [Gregor von Laszewski]
- Adapting the authentication of kilo and modifying the register command to do proper scp. [Gregor von Laszewski]
- Remove insecure option. [Gregor von Laszewski]
- Surpress cert warning. [Gregor von Laszewski]
- Remove the register.host method as it is replaced by register.remote. [Gregor von Laszewski]
- Fixing the yaml file. [Gregor von Laszewski]
- Implementing a working remote register command taht uses the etc file to locate the remote openrc file. [Gregor von Laszewski]
- Change format of cloudmesh yaml ant introduce openrc location. [Gregor von Laszewski]
- Removing india from etc and replacing with kilo and juno. Fixing the etc cloudmesh.yaml for juno and kilo. [Gregor von Laszewski]
- Returning an object in ll instead of printing it. [Gregor von Laszewski]
- Add the manpage for the future comet command. [Gregor von Laszewski]
- Starting a prototype of the comet cluster command. [Gregor von Laszewski]

- Adding a future hpc test cluster comand. [Gregor von Laszewski]
- Adding group attributes for future hpc commands. [Gregor von Laszewski]
- Update of hpc manual page for future commands. [Gregor von Laszewski]
- Adding comet logon, logoff. [Gregor von Laszewski]
- Simple docopts changes. [ehdsouza]
- Modified register list user manual. [ehdsouza]
- Modified a sentence in quota user manual. [ehdsouza]

Fix

- Modified user manual for cm register rc. [ehdsouza]

Other

- Dealing with both v2 and v3 of keystone. [fugangwang]
- Example of working with keystone v3. [fugangwang]
- **chg** usr: moving the import statements of sandman into the command so we do not see the sqlalchemy warning at startup. [Gregor von Laszewski]
- Dev: new: Hpc command for squeue. [ehdsouza]
- Chr: dev: change version to __version__ [Gregor von Laszewski]
- Version 0.7.1. [Gregor von Laszewski]

0.7.1 (2015-09-30)

- Version 0.7.0. [Gregor von Laszewski]

0.7.0 (2015-09-30)

New

- Added rst and nosetests along with minor modifications for quota. [ehdsouza]
- Inventory command. [Gregor von Laszewski]
- Simple docker makefile commands and elementary notes. [Gregor von Laszewski]
- Dockerfile added. [Gregor von Laszewski]
- Add a vm name prototype function. [Gregor von Laszewski]
- Proposal of system preparation instructions for windows. [Gregor von Laszewski]
- Adding models for VM, FLAVOR, IMAGE. [Gregor von Laszewski]
- Adding automatic changelog generation. [Gregor von Laszewski]

Changes

- Changed register command to specify albert and removed two commands from user manual. [ehdsouza]
- Documentation of basic cm command and options. [Gregor von Laszewski]
- Improve the instalation instructions of the source code. [Gregor von Laszewski]
- Adding ubuntu 15.04 instructions. [Gregor von Laszewski]
- Improving CentOS documentation. [Gregor von Laszewski]
- Reorganize the system preparation documentation and adding OSX instructions. [Gregor von Laszewski]
- Remove documentation dependency for cmd3 in the instalation. [Gregor von Laszewski]
- Provide a working install instruction for ssh and git on windows. [Gregor von Laszewski]
- Improving documentation of group command. [Gregor von Laszewski]
- Update system.rst with installation steps for windows. [Gourav Shenoy]

Fix

- Enabled NovaCommand, added nosetests, fixing pyreadline version, fixed pep08 warnings. [Mangirish Wagle]

Other

- **chg** dev: significant comments to the register command. [Gregor von Laszewski]
- Add examples for group command in command_group.rst. [Gourav Shenoy]
- Add notifications for travis. [Gregor von Laszewski]
- Auto index in the commands doc dir. [Gregor von Laszewski]
- Added placeholder foe command documentation. [Gregor von Laszewski]
- Merge. [Gregor von Laszewski]
- Add simple git issue printer. [Gregor von Laszewski]
- Changing the filename to unix style. [Gregor von Laszewski]
- Improved the system windows install instructions. [Gregor von Laszewski]
- **chg** dev: using –format=FORMAT instead of –output. [Gregor von Laszewski]

  commands. [Gregor von Laszewski]
- **fix** usr: improving cloud command documentation. [Gregor von Laszewski]
- Made changes to setup.py so that cloudmesh.yaml is installed at "~/.cloudmesh/cloudmesh.yaml" [Erika Dsouza]
- Dev: Update ChangeLog. [Gregor von Laszewski]
- Use old spelling of SSHCommand. [Gregor von Laszewski]

- Dev: Update ChangeLog. [Gregor von Laszewski]
- Dev: Update ChangeLog. [Gregor von Laszewski]
- Dev: Update ChangeLog. [Gregor von Laszewski]
- Using Config.path_expand from common.ConfigDict. [Gregor von Laszewski]
- Ad travis test for flaten dict. [Gregor von Laszewski]
- Update version. [Gregor von Laszewski]

0.6.9 (2015-09-06)

- Using Config.path_expand from common.ConfigDict. [Gregor von Laszewski]
- Ad travis test for flaten dict. [Gregor von Laszewski]
- Update version. [Gregor von Laszewski]

0.6.8 (2015-09-06)

- Add default test to travis. [Gregor von Laszewski]
- Add user key. [Gregor von Laszewski]
- Making Default set and get work. [Gregor von Laszewski]
- Adding more passing tests to travis. [Gregor von Laszewski]
- Make the config dictests more portable on windows. [Gregor von Laszewski]
- Move some tests to an old dir for later usage. [Gregor von Laszewski]
- Version. [Gregor von Laszewski]

0.6.7 (2015-09-05)

- Travis cleanup. [Gregor von Laszewski]
- Testing doc and cm. [Gregor von Laszewski]
- Force renaming to SecureShellCommand. [Gregor von Laszewski]
- Explicit call of cm via python and not sh. [Gregor von Laszewski]
- Travis experiment. [Gregor von Laszewski]
- Pip install . [Gregor von Laszewski]
- Pip install . [Gregor von Laszewski]
- More travis esxperiments. [Gregor von Laszewski]
- A travis pythonpath set. [Gregor von Laszewski]
- Ignore pep8 warning. [Gregor von Laszewski]
- Modified travis script. [Gregor von Laszewski]
- Use SSHCommand. [Gregor von Laszewski]
- Use old spaleeing of SSHCommand. [Gregor von Laszewski]
- Add doc to travis. [Gregor von Laszewski]

- Cleanup. [Gregor von Laszewski]

0.6.6 (2015-09-04)

- Switchong the version command to use dict printer. [Gregor von Laszewski]
- Adding a version command. [Gregor von Laszewski]
- Adding a build based on pip install -e . [Gregor von Laszewski]
- Remove the osx install in travis. [Gregor von Laszewski]
- Adding the pyreadline install. [Gregor von Laszewski]
- Adding path management for windows. [Gregor von Laszewski]
- Adding more os.pth.join. [Gregor von Laszewski]
- Start to use os.path.join. [Gregor von Laszewski]
- Integrate the model test into travis. [Gregor von Laszewski]
- Trying osx in travis. [Gregor von Laszewski]

0.6.5 (2015-09-04)

- Documentation for table commands in model. [Gregor von Laszewski]
- Automatically finding the tables din model.py. [Gregor von Laszewski]
- Improving table list commands and introducing a nosetest for it. [Gregor von Laszewski]
- Print pip version. [Gregor von Laszewski]
- Remove help call. [Gregor von Laszewski]
- Remove documentation generation from travis. [Gregor von Laszewski]
- Remove ssh command temproarily. [Gregor von Laszewski]
- Rename the sssh command. [Gregor von Laszewski]
- Add .cloudmesh dir. [Gregor von Laszewski]
- Update to the new env script. [Gregor von Laszewski]
- Fixing the https. [Gregor von Laszewski]
- Add .txt ending. [Gregor von Laszewski]
- Update .travis.yml. [Gregor von Laszewski]

  using https in git clone for travis
- Update travis with requirements. [Gregor von Laszewski]
- Travis from source. [Gregor von Laszewski]
- Improved README. [Gregor von Laszewski]
- Improve README. [Gregor von Laszewski]
- Adding the travis file. [Gregor von Laszewski]
- Sandman integration. [Gregor von Laszewski]
- Sandman integration. [Gregor von Laszewski]

- Adding rest via sandman. [Gregor von Laszewski]
- Adding the reservation object to the table list commands. [Gregor von Laszewski]
- Reorganize absolute imports and @command decorator. [Gregor von Laszewski]
- Optimize imports. [Gregor von Laszewski]
- Worked on the TODO list. [Erika Dsouza]
- RESERVATION db object. [Gregor von Laszewski]
- Reservation prototype placeholder class. [Gregor von Laszewski]
- Adding reservation prototype. [Gregor von Laszewski]
- Help help command added. [Gregor von Laszewski]
- Verbatim man pages. [Gregor von Laszewski]
- Case insensitive sorting in the man command. [Gregor von Laszewski]
- Add a browser open command just like in osx. [Gregor von Laszewski]
- Add command topics and new topica help command. [Gregor von Laszewski]
- Add banner and clear commands. [Gregor von Laszewski]

0.6.4 (2015-09-02)

- Adding a simple man command. [Gregor von Laszewski]
- Added default command prototype. [Gregor von Laszewski]
- Added the GROUP type to the table() so that cm info works. [Gregor von Laszewski]
- Add group. [Gregor von Laszewski]
- Adedd select command. [Gregor von Laszewski]
- Trying from Windows 10. [Erika Dsouza]
- Added group proposal. [Gregor von Laszewski]

0.6.3 (2015-09-01)

- Improving imports. [Gregor von Laszewski]

0.6.2 (2015-09-01)

- Pep8 cleanup. [Gregor von Laszewski]
- Add the key prototype command. [Gregor von Laszewski]
- Add key table. [Gregor von Laszewski]
- Cleanup of the table info method. [Gregor von Laszewski]
- Added a reworked model and a simple DEFAULT object as example. [Gregor von Laszewski]
- Simple ssh pass through. [Gregor von Laszewski]
- Adds ssh command template. [Gregor von Laszewski]
- 0.6.1. [Gregor von Laszewski]

0.6.1 (2015-08-31)

- Rename command filenames. [Gregor von Laszewski]
- Added some comments what to do next. [Gregor von Laszewski]
- Changing the syntax at 2 other places. [Mangirish Wagle]
- Changed py files to use format and print() syntax. [Mangirish Wagle]
- Moving first commands to the new shell. [Gregor von Laszewski]
- Start draft of configuration instructions. [Gregor von Laszewski]
- Add some more windows instructions. [Gregor von Laszewski]
- Some documentation improvements for console and windows. [Gregor von Laszewski]
- Add mysql example. [Gregor von Laszewski]
- Add simple nosetest examples. [Gregor von Laszewski]
- Code rules. [Gregor von Laszewski]
- Moving towards format instead of % [Gregor von Laszewski]
- Introducing portable colors between linux and windows. [Gregor von Laszewski]
- Better splash handeling. [Gregor von Laszewski]
- Remove parameter from csv table. [Gregor von Laszewski]
- Delete cmd3.yaml file. [Gregor von Laszewski]
- Tag new version. [Gregor von Laszewski]
- 0.6.0. [Gregor von Laszewski]

0.6.0 (2015-08-29)

- Remove junk file again. [Gregor von Laszewski]
- Cleanup dev dirs. [Gregor von Laszewski]
- Moving dirs. [Gregor von Laszewski]
- Commit python script to start/stop/reboot vm in any cloud env. [Gourav Shenoy]
- Adding scripts to spawn VMs across clouds, including India. [Mangirish Wagle]
- Fixing interactive and non interactive mode. [Gregor von Laszewski]
- Adding the @command decorator to generate functions with docopt parameters. [Gregor von Laszewski]
- Basic cm command. [Gregor von Laszewski]
- Simplify cmd. [Gregor von Laszewski]
- Real simple cmd. [Gregor von Laszewski]
- Fix the formatting. [Gregor von Laszewski]
- Todo faker for larger example. [Gregor von Laszewski]
- Use configdict. [Gregor von Laszewski]
- Introduce classes. [Gregor von Laszewski]
- Tip on using format. [Gregor von Laszewski]

- Pep8. [Gregor von Laszewski]
- Comments to include a new class. [Gregor von Laszewski]
- Remove junk file. [Gregor von Laszewski]
- Moved developer code to dev. [Gregor von Laszewski]
- Ignore emacs backup files. [Gregor von Laszewski]
- Add missing file. [Gregor von Laszewski]
- Remove sh dependency. [Gregor von Laszewski]
- Removing fabric as it does not work in windows. [Gregor von Laszewski]
- Remove the idea file. [Gregor von Laszewski]
- Commit python script to start/stop/reboot VMs. [Gourav Shenoy]
- Adding the persistent dict. [Erika Dsouza]
- Gourav commit with passphrase. [Gourav Shenoy]
- Adding again. [Erika Dsouza]
- Erika added. [Erika Dsouza]
- Commiting README.rst with SSH. [Mangirish Wagle]
- Gourav commit. [Gourav Shenoy]
- Update .gitignore. [Gregor von Laszewski]
- Update .gitignore. [Gregor von Laszewski]
- Update .gitignore. [Gregor von Laszewski]
- Gregor test checkin. [Gregor von Laszewski]
- Changing gitignore. [mangirish]
- Remove fabric as fabric does not work on windows. [Gregor von Laszewski]
- Remove the sh tag dependency. [Gregor von Laszewski]
- Adding comments to the non working windows documentation. [Gregor von Laszewski]
- Update source documentation. [Gregor von Laszewski]
- Add dev docs. [Gregor von Laszewski]
- Update README.rst. [Gregor von Laszewski]
- Documentation framework. [Gregor von Laszewski]
- Remove autogenerated code. [Gregor von Laszewski]
- Position of kwrags. [Gregor von Laszewski]
- Remove pycharm files. [Gregor von Laszewski]
- Update .gitignore. [Gregor von Laszewski]
- Remove debug msg. [Gregor von Laszewski]
- Added group. [Gregor von Laszewski]
- Added priorities. [Gregor von Laszewski]
- Update map. [Gregor von Laszewski]

- Update map. [Gregor von Laszewski]
- Rename. [Gregor von Laszewski]
- Add mindmap and dictdb template. [Gregor von Laszewski]
- Base functions for flavor, image, etc. [Hyungro Lee]
- Merge. [Hyungro Lee]
- Sample yaml file. [Gregor von Laszewski]
- Working on servers. [Hyungro Lee]
- Add and delete keys api. [Gregor von Laszewski]
- Format update. [Hyungro Lee]
- Test exceptions. [Hyungro Lee]
- Typo. [Hyungro Lee]
- Key create. [Gregor von Laszewski]
- Remove debug msg. [Gregor von Laszewski]
- Use quota defaults instead of get by user as permission denied. [Gregor von Laszewski]
- Limits. [Gregor von Laszewski]
- Simplified openstack interface. [Gregor von Laszewski]
- Towards python openstack api. [Gregor von Laszewski]
- Clear the defaults after tests. [Gregor von Laszewski]
- Adding tests for default and Default.clear images flavor group. [Gregor von Laszewski]
- Ad an interactive cloud selector. [Gregor von Laszewski]
- Fixing the default selector for keys. [Gregor von Laszewski]
- Get function for registered cloud. [Gregor von Laszewski]
- Allow force in cloudmesh base yn_choice. [Gregor von Laszewski]
- Replace yn query with existing yn_choice. [Gregor von Laszewski]
- Fixing syntax and various bugs. [Gregor von Laszewski]
- Implement the missing get function. [Gregor von Laszewski]
- Fix the key add function. [Gregor von Laszewski]
- Git add keys. [Gregor von Laszewski]
- Debug. [Gregor von Laszewski]
- Add function was not doing the right thing. [Gregor von Laszewski]
- Start fisrt fix of the db key manager. [Gregor von Laszewski]
- Fixing the list functions in cm key. [Gregor von Laszewski]
- Creating command-register doc. [Daniel Silva]
- Updating test_vm. [Daniel Silva]
- Deleting a set of vm by name - vm delete sample-[1-10] [Daniel Silva]
- Deleting vm by name - vm command. [Daniel Silva]

- Fixing some functions. [Paulo Chagas]
- Implementing vm delete. [Daniel Silva]
- Updating some functions on mesh and sshkeydbmanager. [Paulo Chagas]
- Improving test_vm. [Daniel Silva]
- Fixing circular import. [Paulo Chagas]
- Fixing a bug. [Paulo Chagas]
- Removing a useless function in vm command. [Daniel Silva]
- Command vm start created. [Daniel Silva]
- Updating clear and dump. [Paulo Chagas]
- Removed unresolved reference to cloud/Cloud.py. [Daniel Silva]
- Mesh.clouds implemented. [Paulo Chagas]
- Created vm start command. [Daniel Silva]
- Updating cm key list. [Paulo Chagas]
- Updating get_from_yaml and other needed functions. [Paulo Chagas]
- Adding cloudmesh native provider. [Gregor von Laszewski]
- Introducing mesh. [Gregor von Laszewski]
- Updating key default and key delete. [Paulo Chagas]
- Updating key delete using –select. [Paulo Chagas]
- Fixing key command. [Paulo Chagas]
- Created doc for command_vm functions. [Daniel Silva]
- Created command_vm functions. [Daniel Silva]
- Some updates to key command (almost done) [Paulo Chagas]
- Adding some function to SSHKeyDBManager. [Paulo Chagas]
- Update function included. function usage documentation included. [Paulo Chagas]
- Created tests for nova command. [Daniel Silva]
- –select option implemented. SSKeyDBManager created with some methods. Some tests made. [Paulo Chagas]
- Fixed register tests. [Daniel Silva]
- Updating filename to uri. [Paulo Chagas]
- Adding requirements. [Gregor von Laszewski]
- Added test documentation. [Gregor von Laszewski]
- Adding tests for vm, flavor, image information from an openstack cloud. [Gregor von Laszewski]
- Remove the : from the debug message string. [Gregor von Laszewski]
- Replace log.debug with sanitized self.DEBUG. [Gregor von Laszewski]
- Avoid printing the password in debug mode. [Gregor von Laszewski]
- Adding native openstack interface without libcloud. [Gregor von Laszewski]
- Fixing the key management functions and tests. [Gregor von Laszewski]

- Completing cm admin command. [Gregor von Laszewski]
- Fixed path in host function in CloudRegister.py. [Daniel Silva]
- Split up cm_shell_cloud into cloud, list, register. [Gregor von Laszewski]
- Rmove loglevel command that is already in cmd3. [Gregor von Laszewski]
- Removing try exceptions. [Paulo Chagas]
- Testing run method. [Gregor von Laszewski]
- Doing an improved usint test on register. [Gregor von Laszewski]
- Improved directory function in CloudRegister.py. [Daniel Silva]
- Refactor CloudRegister. [Gregor von Laszewski]
- Pep8. [Gregor von Laszewski]
- Fixing indentation. [Gregor von Laszewski]
- Start refactoring. [Gregor von Laszewski]
- Refactor the SSHkey classes and filenames. [Gregor von Laszewski]
- Editing keys, util and test_keys. [Paulo Chagas]
- Merge. [Gregor von Laszewski]
- Adding key for test. [Paulo Chagas]
- Fixing test_keys.py. [Paulo Chagas]
- Adding __init__.py. [Paulo Chagas]
- Updating test_keys. [Paulo Chagas]
- Updating test_keys. [Paulo Chagas]
- Updating test_keys. [Paulo Chagas]
- Updating documentation. [Paulo Chagas]
- Improve Key Management. [Gregor von Laszewski]
- Rename clas sto SSHkey. [Gregor von Laszewski]
- Adding SSHkey type derived from old ssh key cloudmesh key.util.py. [Gregor von Laszewski]
- Editing “if”s on limits command. [Paulo Chagas]
- Editing “if”s on quota command. [Paulo Chagas]
- Editing “if”s on ssh command. [Paulo Chagas]
- Getting geys from .ssh and github. [Gregor von Laszewski]
- Improve the new-ENV script. [Gregor von Laszewski]
- Documentation management changes for changes and authors. [Gregor von Laszewski]
- Changes in command cloud - register-dir function. [Daniel Silva]

0.5.8 (2015-06-28)

- Add bin commands to create a new environment from the commandline. [Gregor von Laszewski]
- Move md to rst. [Gregor von Laszewski]
- Adding autoinstall. [Gregor von Laszewski]
- Rename README.rst to README.md. [Gregor von Laszewski]
- Update README.rst. [Gregor von Laszewski]
- Created register_CERT command in command_cloud.py. [Daniel Silva]
- Editing "if"s on volume command. [Paulo Chagas]
- Created command line arguments in cm_shell_[admin, key, loglevel and status] [Daniel Silva]
- Editing "if"s on vm command. The –command has some issue: docopt separate commands by space. [Paulo Chagas]
- Adding the home of hacking and CONTRIBUTING. [Gregor von Laszewski]
- Updare vm command. [Gregor von Laszewski]
- Cluster command. [Hyungro Lee]
- Indentation. [Hyungro Lee]
- Indentation. [Hyungro Lee]
- Launcher commands with examples and description. [Hyungro Lee]
- Stack update. [Hyungro Lee]
- Editing "if"s on secgroup command. [Paulo Chagas]
- Created register india command. [Daniel Silva]
- Editing "if"s on refresh command. [Paulo Chagas]
- Adding selector. [Gregor von Laszewski]
- Adding a select command. [Gregor von Laszewski]
- Finalizing the refersh command. [Gregor von Laszewski]
- Assignments. [Gregor von Laszewski]
- Editing volume. [Paulo Chagas]
- Editing volume. [Paulo Chagas]
- Editing volume. [Paulo Chagas]
- Updated register india command. [Daniel Silva]
- Updated cluster section in old.rst. [Daniel Silva]
- Editing volume. [Paulo Chagas]
- Editing volume documentation. [Paulo Chagas]
- Merge. [Gregor von Laszewski]
- Use full name of india to avoid dependency of ssh config. [fugangwang]
- 79 character. [Gregor von Laszewski]
- Remove use of < > [Gregor von Laszewski]

- Merge. [Gregor von Laszewski]
- Merge. [Gregor von Laszewski]
- Security group docopt; urllib3 in requirements. [fugangwang]
- Improve vm command. [Gregor von Laszewski]
- Fix list commnd. [Gregor von Laszewski]
- Updating cm_shell_cloud. [Paulo Chagas]
- Removing file. [Paulo Chagas]
- Fixing key. [Paulo Chagas]
- Adding commands. [Paulo Chagas]
- Improve documentation. [Gregor von Laszewski]
- Add refresh command. [Gregor von Laszewski]
- Adding the missing rst docments. [Gregor von Laszewski]
- Adding a template. [Gregor von Laszewski]
- Fix verbatim vormatting of commands. [Gregor von Laszewski]
- Just adding import os. [Paulo Chagas]
- Finishing cm search. It is possible to filter with =,!=,<,<=,>,>=. Test for cm search was added. [Paulo Chagas]
- Improving default and adding old manual pages. [Gregor von Laszewski]
- Summary change file. [Gregor von Laszewski]
- Adding a delete function. [Gregor von Laszewski]
- Fixing the use of Shell.scp. [Gregor von Laszewski]
- Removing debug messages. [Gregor von Laszewski]
- Just adding import getpass. [Paulo Chagas]
- Fixing the default command. [Gregor von Laszewski]
- Fixed search command. [Paulo Chagas]
- Improving documentation. [Gregor von Laszewski]
- Adding the set and get default methods. [Gregor von Laszewski]
- Adding a default get command. [Gregor von Laszewski]
- Add sample search. [Gregor von Laszewski]
- Test register_CERT. [Daniel Silva]
- Simple correction on how to use the os.system command. [Gregor von Laszewski]
- Add todos for instalation instructions. [Gregor von Laszewski]
- Changes in command register_CERT. [Daniel Silva]
- Adding documentation. [Gregor von Laszewski]
- Updating some of the command ideas. [Gregor von Laszewski]
- Use code. [Gregor von Laszewski]
- Improving simple api documentation. [Gregor von Laszewski]

- First api documentation to use dicts for change. [Gregor von Laszewski]
- Update from dict. [Gregor von Laszewski]
- Simplify tests. [Gregor von Laszewski]
- Adding a uniqe cm_id and cm_type. [Gregor von Laszewski]
- Created cm register CLOUD CERT command. [Daniel Silva]
- Updating search and default command. [Paulo Chagas]
- Filter for entries like [001-002] [Paulo Chagas]
- Created cm register india command. [Daniel Silva]
- Fix the module path. [Gregor von Laszewski]
- Remove unneeded files. [Gregor von Laszewski]
- Moving files into final position. [Gregor von Laszewski]
- Editing filter to get more than one option (AND) [Paulo Chagas]
- Adding a try-except for invalid search. [Paulo Chagas]
- Adding test if result query is empty. [Paulo Chagas]
- Test add command register test. [Daniel Silva]
- Updated on the examples for aws and azure. [Hyungro Lee]
- Outstanding checkins. [Gregor von Laszewski]
- Removing None from table print. [Gregor von Laszewski]
- Implementing a first cm list command. [Gregor von Laszewski]
- Removed plural of flavor, vm, image in kind strings. [Gregor von Laszewski]
- Start reorganizing methods. [Gregor von Laszewski]
- Splitting classes. [Gregor von Laszewski]
- Start baseclass. [Gregor von Laszewski]
- Simple code cleanip. [Gregor von Laszewski]
- Assert Parameter.expand test. [Gregor von Laszewski]
- Pep8, switching hosstlist to Parameter.expand. [Gregor von Laszewski]
- Adding more comments. [Gregor von Laszewski]
- Improving the hostlist comment. [Gregor von Laszewski]
- Adding more command suggestions. [Gregor von Laszewski]
- Reformatting, reordering, and sellchecking the proposed commands. [Gregor von Laszewski]
- Adding cloudmesh_search. [Paulo Chagas]
- Remove debug messages. [Gregor von Laszewski]
- Adding a test file to queries. [Paulo Chagas]
- Adding a test file for queries. [Paulo Chagas]
- Renaming the id to cm_id so it does not conflict with the SQL id. [Gregor von Laszewski]
- Fixed commands_nova.rst format. [Daniel Silva]

- Change table data types. [Gregor von Laszewski]
- Updated suggestion commands_nova.rst. [Daniel Silva]
- Adding the export line match. [Gregor von Laszewski]
- Fix the plugin location. [Gregor von Laszewski]
- Libcloud install restored. [Hyungro Lee]
- Debug messages. [Gregor von Laszewski]
- Added __init__ [Gregor von Laszewski]
- Updated suggestion command nova doc. [Daniel Silva]
- Passing git+ strings to pbr does not work. [Gregor von Laszewski]
- Azure tested to create and delete a vm. [Hyungro Lee]
- Default image for azure. [Hyungro Lee]
- Created file suggestion commands nova in doc. [Daniel Silva]
- Fixing the module import. [Gregor von Laszewski]
- Changing more imports. [Gregor von Laszewski]
- Cahnge import statements. [Gregor von Laszewski]
- Reorganize the module. [Gregor von Laszewski]
- Dev version of libcloud from github 'trunk' [Hyungro Lee]
- Reorganize the modules. [Gregor von Laszewski]
- Preparing for update on vm and image. [Gregor von Laszewski]
- Abstracting the lister and inserter. [Gregor von Laszewski]
- Removing one test. [Gregor von Laszewski]
- Reduce code duplication. [Gregor von Laszewski]
- Update search functions to convert to dicts. [Gregor von Laszewski]
- Working on filtered get. [Gregor von Laszewski]
- Readding the deleted methods during a merge. [Gregor von Laszewski]
- Redoing the merge. [Gregor von Laszewski]
- Doc: updated to cloudmesh_cloud. [Paulo Chagas]
- Updated on azure. [Hyungro Lee]
- Azure example started. [Hyungro Lee]
- Doc: Changing cloudmesh_client for cloudmesh_cloud. [Paulo Chagas]
- Testing git commit on windows. [Paulo Chagas]
- Testing git commit on windows. [Paulo Chagas]
- Added cmd3.yaml example to etc directory. [Paulo Chagas]
- Rename. [Hyungro Lee]
- Aws example. [Hyungro Lee]
- Adding a simplified boot command. [Gregor von Laszewski]

- Adding cm_user cm_cloud cm_update attributes to the dicts. [Gregor von Laszewski]
- Adding provider list queris for openstack. [Gregor von Laszewski]
- Adding IMAGE and FLAVOR to get_kind_from_string. [Gregor von Laszewski]
- Adding the username to the vm name for better debugging. [Gregor von Laszewski]
- Adding proper argument pass through to the nova command. [Gregor von Laszewski]
- Assing more objects to the database. [Gregor von Laszewski]
- Adding database replace, add, and merge. [Gregor von Laszewski]
- Adding additional database methods. [Gregor von Laszewski]
- Next_name() [Gregor von Laszewski]
- Renaming the examples to avoid name conflict with existing package. [Gregor von Laszewski]
- Doc: updated to cloudmesh_cloud. [Paulo Chagas]
- Updated on azure. [Hyungro Lee]
- Azure example started. [Hyungro Lee]
- Doc: Changing cloudmesh_client for cloudmesh_cloud. [Paulo Chagas]
- Testing git commit on windows. [Paulo Chagas]
- Testing git commit on windows. [Paulo Chagas]
- Added cmd3.yaml example to etc directory. [Paulo Chagas]
- Rename. [Hyungro Lee]
- Aws example. [Hyungro Lee]
- Pushing first data to the database. [Gregor von Laszewski]
- Adding documentation to the flatten methods. [Gregor von Laszewski]
- Flatten methods for libcloud image and vm. [Gregor von Laszewski]
- F = flatten(d) # function to flatten a dict. [Gregor von Laszewski]
- Adding a simplified boot command. [Gregor von Laszewski]
- Adding cm_user cm_cloud cm_update attributes to the dicts. [Gregor von Laszewski]
- Adding provider list queris for openstack. [Gregor von Laszewski]
- Adding IMAGE and FLAVOR to get_kind_from_string. [Gregor von Laszewski]
- Adding the username to the vm name for better debugging. [Gregor von Laszewski]
- Adding proper argument pass through to the nova command. [Gregor von Laszewski]
- Assing more objects to the database. [Gregor von Laszewski]
- Adding database replace, add, and merge. [Gregor von Laszewski]
- Adding additional database methods. [Gregor von Laszewski]
- Next_name() [Gregor von Laszewski]
- Renaming the examples to avoid name conflict with existing package. [Gregor von Laszewski]
- Fixed the nosetests for table printers. [Gregor von Laszewski]
- Update the Changelog. [Gregor von Laszewski]

- Renaming the files for better readability. [Gregor von Laszewski]
- Moving examples to the examples dir. [Gregor von Laszewski]
- Moving examples to the example dir. [Gregor von Laszewski]
- Fixed test_005 in test_configdict. [Daniel Silva]
- Reorganizing the database code. [Gregor von Laszewski]
- Register default command. [Gregor von Laszewski]
- Reorganizing the cloudmesh_default directory. [Gregor von Laszewski]
- Improving the test dict. [Gregor von Laszewski]
- Add a sample dict for test printing. [Gregor von Laszewski]
- Created Configdict test. [Daniel Silva]
- Adding parameter parsing. [Gregor von Laszewski]
- Format the input to tt to make it more visible in the text. [Gregor von Laszewski]
- Adding cygwin documentation. [Gregor von Laszewski]
- Documentation for next_name. [Gregor von Laszewski]
- Added automatic vm name generator. [Gregor von Laszewski]
- Adding the test method templates for tables. [Gregor von Laszewski]
- Table test template. [Gregor von Laszewski]
- Added testing methods to be implemented. [Gregor von Laszewski]
- Spellchecking. [Gregor von Laszewski]
- Pep8 cleanup; Cloud object template. [Gregor von Laszewski]
- Adding a mailmap. [Gregor von Laszewski]
- Add make view. [Gregor von Laszewski]
- Pep8 cleanup. [Gregor von Laszewski]
- Add developer notes. [Gregor von Laszewski]
- Tag management via makefile. [Gregor von Laszewski]
- Makefile for make doc. [Gregor von Laszewski]
- Adding sphinx documentation framework. [Gregor von Laszewski]
- Simplify CHangelog. [Gregor von Laszewski]
- Clean requirements. [Gregor von Laszewski]
- Add nosetests to tox. [Gregor von Laszewski]
- Imporved documentation. [Gregor von Laszewski]
- Fixing the nova command fixing libcoud example be using path_expand removing timeparse from requirements. [Gregor von Laszewski]
- Changed table.py doc. [Daniel Silva]
- Created command_cloud.py doc. [Daniel Silva]
- Updated ConfigDict.py doc. [Daniel Silva]

0.5.7 (2015-06-17)

- Fixed the nosetests for table printers. [Gregor von Laszewski]
- Update the Changelog. [Gregor von Laszewski]

0.5.6 (2015-06-17)

- Renaming the files for better readability. [Gregor von Laszewski]
- Moving examples to the examples dir. [Gregor von Laszewski]
- Moving examples to the example dir. [Gregor von Laszewski]
- Fixed test_005 in test_configdict. [Daniel Silva]
- Reorganizing the database code. [Gregor von Laszewski]

0.5.5 (2015-06-17)

- Register default command. [Gregor von Laszewski]
- Reorganizing the cloudmesh_default directory. [Gregor von Laszewski]
- Improving the test dict. [Gregor von Laszewski]
- Add a sample dict for test printing. [Gregor von Laszewski]
- Created Configdict test. [Daniel Silva]
- Adding parameter parsing. [Gregor von Laszewski]
- Format the input to tt to make it more visible in the text. [Gregor von Laszewski]
- Adding cygwin documentation. [Gregor von Laszewski]
- Documentation for next_name. [Gregor von Laszewski]
- Added automatic vm name generator. [Gregor von Laszewski]
- Adding the test method templates for tables. [Gregor von Laszewski]
- Table test template. [Gregor von Laszewski]
- Added testing methods to be implemented. [Gregor von Laszewski]
- Spellchecking. [Gregor von Laszewski]
- Pep8 cleanup; Cloud object template. [Gregor von Laszewski]
- Adding a mailmap. [Gregor von Laszewski]
- Add make view. [Gregor von Laszewski]

0.5.4 (2015-06-17)

- Pep8 cleanup. [Gregor von Laszewski]
- Add developer notes. [Gregor von Laszewski]
- Tag management via makefile. [Gregor von Laszewski]
- Makefile for make doc. [Gregor von Laszewski]

- Adding sphinx documentation framework. [Gregor von Laszewski]
- Simplify CHangelog. [Gregor von Laszewski]
- Clean requirements. [Gregor von Laszewski]
- Add nosetests to tox. [Gregor von Laszewski]
- Imporved documentation. [Gregor von Laszewski]
- Fixing the nova command fixing libcoud example be using path_expand removing timeparse from requirements. [Gregor von Laszewski]
- Changed table.py doc. [Daniel Silva]
- Created command_cloud.py doc. [Daniel Silva]
- Updated ConfigDict.py doc. [Daniel Silva]
- Simple tests for configdict. [Gregor von Laszewski]
- Tests created. [Daniel Silva]
- Add tests template. [Gregor von Laszewski]
- Rtd theme. [Gregor von Laszewski]
- Adding sphinx placeholder. [Gregor von Laszewski]
- Add comment placeholder. [Gregor von Laszewski]
- Method needed cls. [Gregor von Laszewski]
- Remove ruamel as it doe snot work well on cygwin. [Gregor von Laszewski]
- Remove ruamel. [Gregor von Laszewski]
- Introduce a Config class. [Gregor von Laszewski]
- Adding not implemented errors. [Gregor von Laszewski]
- Simple tox.ini. [Gregor von Laszewski]
- Adding cm register list ssh command to list the hosts defined in .ssh/config. [Gregor von Laszewski]
- Added registration documentation. [Gregor von Laszewski]
- Added cm register rc host openrc. [Gregor von Laszewski]
- Checks in registration command. [Gregor von Laszewski]
- Uncomment the profile editor. [Gregor von Laszewski]
- Adding a simple form for openstack credentials. [Gregor von Laszewski]
- Fix the register command. [Gregor von Laszewski]
- Dealing with public ip. [fugangwang]
- Cm register commands. [Daniel Silva]
- Added libcloud example; added more requirements. [fugangwang]
- Reorganizing. [Gregor von Laszewski]
- Reorganization. [Gregor von Laszewski]
- Adding first commands. [Gregor von Laszewski]
- Remove version so we use git tags. [Gregor von Laszewski]

- Testing pbr. [Gregor von Laszewski]
- Testing new version of pbr. [Gregor von Laszewski]
- Simple default command. [Gregor von Laszewski]
- Setup. [Gregor von Laszewski]
- Print() [Gregor von Laszewski]
- Move confidict. [Gregor von Laszewski]
- Simple dict printer with corrections. [Gregor von Laszewski]
- Added defaults. [Gregor von Laszewski]
- Introducing os.path.sep instead of / [Gregor von Laszewski]
- Set test. [Gregor von Laszewski]
- Simple configdict based on ruaml.yaml. [Gregor von Laszewski]
- New config dict. [Gregor von Laszewski]
- Initial commit. [Gregor von Laszewski]

12.3 ToDos

Todo

reformat the inventory section to be a real manual.

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/cloudmesh-client/checkouts/stable/docs/source/commands/command_inventory.rst, line 6.)

Todo

verify if this works

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/cloudmesh-client/checkouts/stable/docs/source/commands/command_register.rst, line 34.)

Todo

at this time we have not integrated our AWS and Azure IaaS abstractions in the new cloudmesh client. We will make them available in future.

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/cloudmesh-client/checkouts/stable/docs/source/commands/command_register.rst, line 56.)

Todo

the description of what this is doing was ambigous, we need to clarify if it only replaces to do or actually add things that do not exist, or just overwrites.

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/cloudmesh-client/checkouts/stable/docs/source/commands/command_register.rst, line 210.)

Todo

We used to have a .bak.# when we modified the yaml file, do you still have this

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/cloudmesh-client/checkouts/stable/docs/source/commands/command_register.rst, line 220.)

Todo

setting a default format via defaults

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/cloudmesh-client/checkouts/stable/docs/source/commands/command_templates.rst, line 26.)

Todo

put link to registration here

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/cloudmesh-client/checkouts/stable/docs/source/commands/command_templates.rst, line 60.)

Todo

not yet tested but should work. add cloud registration here

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/cloudmesh-client/checkouts/stable/docs/source/configuration.rst, line 334.)

Todo

not yet supported but used to be so we work on it ASAP. add cloud registration here

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/cloudmesh-client/checkouts/stable/docs/source/configuration.rst, line 339.)

Todo

not yet supported but used to be so we work on it ASAP. add cloud registration here

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/cloudmesh-client/checkouts/stable/docs/source/configuration.rst, line 344.)

Todo

not tested, but should work as is regular openstack. add cloud registration here

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/cloudmesh-client/checkouts/stable/docs/source/configuration.rst, line 349.)

Todo

not yet supported. add cloud registration here

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/cloudmesh-client/checkouts/stable/docs/source/configuration.rst, line 354.)

Todo

This section is incomplete and we need to make sure that tox works. We also need to explain how travis works and how we can run nosetests locally

---

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/cloudmesh-client/checkouts/stable/docs/source/installation.rst, line 73.)

Todo

check if this is the right way to do so

(The *original entry* is located in /home/docs/checkouts/readthedocs.org/user_builds/cloudmesh-client/checkouts/stable/docs/source/system-cygwin.rst, line 134.)

CHAPTER 13

Docker

This is an experimental effort with little documentation

Start a virtual machine that runs docker in it:

```
make docker-machine
```

The machine is called cloudmesh, do not confuse this with the docker image that is bing created and is also called cloudmesh.

Login to the started vm so you can execute docker commands:

```
make docker-machine-login
```

Create the cloudmesh docker image with the name 'cloudmesh':

```
make docker-build
```

Publish the image on docker hub (only Gregor):

```
make docker-login

make docker-publish
```

Get the image (does not work):

```
make docker-pull
```

Not working or incomplete:

```
make docker-run
make docker-clean-images
```

CHAPTER 14

Existing original cloudmesh commands

This section includes the list of old cloudmesh commands. We provide this list here as inspiration and for discussion. The finalized commands will be in the "Commands" section.

Notes:

- –format should become –output? reasoning in api format is keyword.

In case we keep format we want to do output_format as api variable.

currently we have

- list flavors and flavor list, ... maybe not needed and if the one should call just the other
- –refresh in list seems useful
- we need a refresh command that just refreshes and does not list
- we need to urgently work on defaults for list columns
- we need to work on vm/boot command.

14.1 Parametrized commands

Many commands take a number of parameters. Some of thes parameters can be easily referring to a number of different objects, while sing a parametrized syntax when specifying the command.

For example when using the string "machine[01-03]" in a parametrized argument we refer to machine01, machine02, machine03. As you can see it is a simple way of expressing arrays as part of a string. Additionally you can use , to add to it. Thus machine[01-03],machine[06-07] refers to the machines machine01, machine02, machine03, machine06, machine07. This flexible mechanism of specifying parameters is followed consistently within cloudmesh. We indicate in a manual page if a parameter is parametrized by including (parametrized) in its definition of the arguments

In addition we utilize in some cases also the PARAMETER... notation meaning that the parameter can be specified multiple times separated by a space. This is well known from many Linux commandline clients.

14.2 Refresh

```
Refreshes the database with information from the clouds


Usage:
    refresh
```

```
    refresh status
    refresh list
    refresh CLOUD...

Arguments:

    CLOUD  (parametrized) the name of a cloud

Description:

    Refreshes are activated on all clouds that are "active". A cloud
    can be activated with the cloud command

       cloud activate CLOUD

    refresh
        refreshes the information that we have about all
        activeclouds.

    refresh CLOUD...
        refreshes the information form the specific clouds

    refresh status
        as the refresh may be done asynchronously, the stats will
        show you the progress of the ongoing refresh NOT
        IMPLEMENTED It also shows when the last refresh on a
        specific cloud object took place.

    refresh list
        lists all the Clouds that need a refresh

 Example:

     The folowing command sequences each refresh the clouds named
     india and aws.

         refresh india,aws

         refresh india aws

         refresh india
         refresh aws

      To utilize the refresh command without parameters you need to
      asuse the clouds are activated

         cloud activate india
         cloud activate aws
         refresh
```

14.3 Select

```
Select is a command that allows the interactive selection of an item
specified
```

```
Usage:
    select LIST...
    select image CLOUD
    select flavor CLOUD
    select vm CLOUD
    select cloud

Arguments:
    LIST   (parametrized) List of items to choose from
    CLOUD  a single cloudname ti identify from which we select the
           information

Description:
    Returns either None or the item that is chosen interactively

Example:
    select cloud
        will list all available couds and you can interactively
        select one. The name of the selected cloud is returned.
```

14.4 List

```
Usage:
    list flavor [CLOUD...|--all]
                [--refresh]
                [--detail|--columns=COLUMNS]
                [--format=FORMAT]
    list image [CLOUD...|--all]
                [--refresh]
                [--detail|--columns=COLUMNS]
                [--format=FORMAT]
    list vm [CLOUD...|--all]
            [--refresh]
            [--group=GROUP]
            [--detail|--columns=COLUMNS]
            [--format=FORMAT]
    list default [CLOUD...|--all]
                 [--detail|--columns=COLUMNS]
                 [--format=FORMAT]
    list cloud [CLOUD...|--all]
                 [--detail|--columns=COLUMNS]
                 [--format=FORMAT]

Arguments:

    CLOUD...    (parametrized) the names of the clouds for
                which we want to obtain a list,  e.g. india. If
                no cloud name is provided the default cold is
                used. If instead --all is used all active clouds
                are used. A default cloud an be set with

                   default cloud CLOUD

                If install activated clouds
                ar used. If the cloud is not specifies
```

```
Options:

    --refresh               refresh data before the list is refreshed
    --group=GROUP           give the group name in list vm
    --detail                for table print format, with all
                            information. This may however be
                            difficult to read as a lot of
                            information may be returned. If the
                            parameter is omitted a small subset
                            is printed. The columns can be defined
                            with

                               default CLOUD [image|flavor|vm] COLUMNS

                            This can be overwritten by specifying
                            explicit columns
    --columns=COLUMNS       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). If the
                            columns parameter is used a single
                            table is returned. If not a table is
                            printed for each cloud.

    --format=FORMAT         output format: table, json, csv, dict
                            [default: table]

Description:

    List available flavors, images, vms, projects and clouds If
    the CLOUD name is not specified, the default cloud will
    be used. You can interactively set the default cloud with
    the command '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

    If no cloud is specified it lists the information for all clouds.

Examples:

    list flavor india aws
       lists the cloud flavors for india and aws. Two different
       tables are returned
```

```
    list flavor india aws --detail
       lists the cloud flavors for india and aws with lots of
       details. Two different tables are returned.

    list flavor india aws --columns=cloud,name,cm_id
       lists the cloud flavors for india and aws with the
       cloudname, the name of the flavor, and the unique
       cloudmesh id for this flavor. A single table is returned.

See Also:

    cloud help
    cloud activate CLOUD
```

14.5 Security group

```
Usage:
    secgroup list CLOUD TENANT
    secgroup create CLOUD TENANT LABEL
    secgroup delete CLOUD TENANT LABEL
    secgroup rules-list CLOUD TENANT LABEL
    secgroup rules-add CLOUD TENANT LABEL FROMPORT TOPORT PROTOCOL CIDR
    secgroup rules-delete CLOUD TENANT LABEL FROMPORT TOPORT PROTOCOL CIDR
    secgroup -h | --help
    secgroup --version

Options:
    -h            help message

Arguments:
    CLOUD         Name of the IaaS cloud e.g. india_openstack_grizzly.
    TENANT        Name of the tenant, e.g. fg82.
    LABEL         The label/name of the security group
    FROMPORT      Staring port of the rule, e.g. 22
    TOPORT        Ending port of the rule, e.g. 22
    PROTOCOL      Protocol applied, e.g. TCP,UDP,ICMP
    CIDR          IP address range in CIDR format, e.g., 129.79.0.0/16

Description:
    security_group command provides list/add/delete
    security_groups for a tenant of a cloud, as well as
    list/add/delete of rules for a security group from a
    specified cloud and tenant.


Examples:
    $ secgroup list india fg82
    $ secgroup rules-list india fg82 default
    $ secgroup create india fg82 webservice
    $ secgroup rues-add india fg82 webservice 8080 8088 TCP "129.79.0.0/16"
```

14.6 Cloud

```
Usage:
    cloud refresh
    cloud list [CLOUD...] [--refresh] [--columns=COLUMNS] [--format=FORMAT] [--details]
    cloud alias NAME [CLOUD]
    cloud on [CLOUD...]
    cloud off [CLOUD...]
    cloud TODO add YAMLFILE [--force] REMOVE_REPLACED_BY_REGISTER
    cloud TODO remove [CLOUD|--all]   MOVE_TO_REGISTER
    cloud default
    cloud default CLOUD
    cloud set flavor [CLOUD] [--name=NAME|--id=ID]
    cloud set image [CLOUD] [--name=NAME|--id=ID]

TODO: aad the selector

Arguments:

  CLOUD                  the name of a cloud
  YAMLFILE               a yaml file (with full file path) containing
                         cloud information
  NAME                   name for a cloud (or flavor and image)

Options:

   --columns=COLUMNS     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,
                         semiall to display all except credentials
                         and defaults)

   --format=FORMAT       output format: table, json, csv

   --all                 display all available columns

   --force               if same cloud exists in database, it will be
                         overwritten

   --name=NAME           provide flavor or image name

   --id=ID               provide flavor or image id


Description:

    TODO fix the description

    The cloud command allows easy management of clouds in the
    command shell. The following subcommands exist:

    cloud [list] [--column=COLUMN] [--json|--table]
        lists the stored clouds, optionally, specify columns
        for more cloud information. For
```

```
        example, --column=active,label

    cloud info [CLOUD|--all] [--json|--table]
        provides the available information about the cloud in dict
        format
        options: specify CLOUD to display it, --all to display all,
                 otherwise selected cloud will be used

    cloud alias NAME [CLOUD]
        sets a new name for a cloud
        options: CLOUD is the original label of the cloud, if
                 it is not specified the default cloud is used.


    cloud select [CLOUD]
        selects a cloud to work with from a list of clouds. If
        the cloud is not specified, it asks for the cloud
        interactively

    cloud on [CLOUD]
    cloud off [CLOUD]
        activates or deactivates a cloud. if CLOUD is not
        given, the default cloud will be used.


    cloud add YAMLFILE [--force]
        adds the cloud information to database that is
        specified in the YAMLFILE. This file is a yaml. You
        need to specify the full path. Inside the yaml, a
        cloud is specified as follows:

        cloudmesh:
           clouds:
             cloud1: ...
             cloud2: ...

        For examples on how to specify the clouds, please see
        cloudmesh.yaml

        options: --force. By default, existing cloud in
                 database cannot be overwritten, the --force
                 allows overwriting the database values.

    cloud remove [CLOUD|--all]
        remove a cloud from the database, The default cloud is
        used if CLOUD is not specified.
        This command should be used with caution. It is also
        possible to remove all clouds with the option --all

    cloud default [CLOUD|--all]

        show default settings of a cloud, --all to show all clouds

    cloud set flavor [CLOUD] [--name=NAME|--id=ID]

        sets the default flavor for a cloud. If the cloud is
        not specified, it used the default cloud.
```

```
    cloud set image [CLOUD] [--name=NAME|--id=ID]

        sets the default flavor for a cloud. If the cloud is
        not specified, it used the default cloud.
```

14.7 VM

```
Usage:
    vm start [--name=NAME]
             [--count=COUNT]
             [--cloud=CLOUD]
             [--image=IMAGE_OR_ID]
             [--flavor=FLAVOR_OR_ID]
             [--group=GROUP]
    vm delete [NAME_OR_ID...]
              [--group=GROUP]
              [--cloud=CLOUD]
              [--force]
    vm ip assign [NAME_OR_ID...]
                 [--cloud=CLOUD]
    vm ip show [NAME_OR_ID...]
               [--group=GROUP]
               [--cloud=CLOUD]
               [--format=FORMAT]
               [--refresh]
    vm login NAME [--user=USER]
             [--ip=IP]
             [--cloud=CLOUD]
             [--key=KEY]
             [--] [COMMAND...]
    vm list [CLOUD|--all]
            [--group=GROUP]
            [--refresh]
            [--format=FORMAT]
            [--columns=COLUMNS|--deatil]

Arguments: COMMAND   positional arguments, the commands
                     you want to execute on the server
                     (e.g. ls -a), you will get a return
                     of executing result instead of login
                     to the server, note that type in --
                     is suggested before you input the
                     commands

    NAME_OR_ID  (parametrized for delete) server name or
                id
    CLOUD       (parametrized for list) the name of the
                cloud. If not specified the deafult clod
                will be used
    KEY         the name of the key to be used at login.
    FORMAT      the format
    COLUMNS     the list of columns
    GROUP       the group name

Options:
```

```
    --columns=COLUMNS      specify what information to display in
                           the columns of the list command.

    --format=FORMAT        output format: table, json, csv, dict
                           [default: table]

    --ip=IP          give the public ip of the server
    --cloud=CLOUD    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
    --detail         for table print format, a brief version
                     is used as default, use this flag to print
                     detailed table
    --flavor=FLAVOR_OR_ID  give the name or id of the flavor
    --group=GROUP          give the group name of server
    --image=IMAGE_OR_ID    give the name or id of the image
    --key=KEY        spicfy a key to use, input a string which
                     is the full path to the public key
                     file [deafult: ~/.ssh/id_rsa.pb]
    --user=USER      give the user name of the server that you want
                     to use to login
    --name=NAME      give the name of the virtual machine
    --force          delete vms without user's confirmation



Description:
    commands used to start or delete servers of a cloud

    vm start [options...]       start servers of a cloud, user may specify
                                flavor, image .etc, otherwise default values
                                will be used, see how to set default values
                                of a cloud: cloud help
    vm delete [options...]      delete servers of a cloud, user may delete
                                a server by its name or id, delete servers
                                of a group or servers of a cloud, give prefix
                                and/or range to find servers by their names.
                                Or user may specify more options to narrow
                                the search
    vm ip assign [options...]   assign a public ip to a VM of a cloud
    vm ip show [options...]     show the ips of VMs
    vm login [options...]       login to a server or execute commands on it
    vm list [options...]        same as command "list vm", please refer to it

Tip:
    in some cases the VM name is parameterized which is very
    convenient when you need a range of VMs e.g. sample[1-3]
    => ['sample1', 'sample2', 'sample3']
    sample[1-3,18] => ['sample1', 'sample2', 'sample3', 'sample18']

Examples:
    vm start --count=5 --group=test --cloud=india
            start 5 servers on india and give them group
            name: test

    vm delete --group=test --names=sample_[1-9]
            delete servers on selected or default cloud with search conditions:
            group name is test and the VM names are among sample_1 ... sample_9
```

```
    vm ip show --names=sample_[1-5,9] --format=json
            show the ips of VM names among sample_1 ... sample_5 and sample_9 in
            json format
```

14.8 Volume

```
Usage:
    volume list
    volume create SIZE
                  [--snapshot-id=SNAPSHOT-ID]
                  [--image-id=IMAGE-ID]
                  [--display-name=DISPLAY-NAME]
                  [--display-description=DISPLAY-DESCRIPTION]
                  [--volume-type=VOLUME-TYPE]
                  [--availability-zone=AVAILABILITY-ZONE]
    volume delete VOLUME
    volume attach SERVER VOLUME DEVICE
    volume detach SERVER VOLUME
    volume show VOLUME
    volume SNAPSHOT-LIST
    volume snapshot-create VOLUME-ID
                           [--force]
                           [--display-name=DISPLAY-NAME]
                           [--display-description=DISPLAY-DESCRIPTION]
    volume snapshot-delete SNAPSHOT
    volume snapshot-show SNAPSHOT
    volume help


volume management

Arguments:
    SIZE              Size of volume in GB
    VOLUME            Name or ID of the volume to delete
    VOLUME-ID         ID of the volume to snapshot
    SERVER            Name or ID of server(VM).
    DEVICE            Name of the device e.g. /dev/vdb. Use "auto" for
                      autoassign (if supported)
    SNAPSHOT          Name or ID of the snapshot

Options:
    --snapshot-id SNAPSHOT-ID     Optional snapshot id to create
                                  the volume from.  (Default=None)
    --image-id IMAGE-ID           Optional image id to create the
                                  volume from.  (Default=None)
    --display-name DISPLAY-NAME   Optional volume name. (Default=None)
    --display-description DISPLAY-DESCRIPTION
                                  Optional volume description. (Default=None)
    --volume-type VOLUME-TYPE
                                  Optional volume type. (Default=None)
    --availability-zone AVAILABILITY-ZONE
                                  Optional Availability Zone for
                                  volume. (Default=None)
    --force                       Optional flag to indicate whether to snapshot a
                                  volume even if its
```

```
                                    attached to an
                                    instance. (Default=False)

Description:
    volume list
        List all the volumes
    volume create SIZE [options...]
        Add a new volume
    volume delete VOLUME
        Remove a volume
    volume attach SERVER VOLUME DEVICE
        Attach a volume to a server
    volume-detach SERVER VOLUME
        Detach a volume from a server
    volume show VOLUME
        Show details about a volume
    volume snapshot-list
        List all the snapshots
    volume snapshot-create VOLUME-ID [options...]
        Add a new snapshot
    volume snapshot-delete SNAPSHOT
        Remove a snapshot
    volume-snapshot-show SNAPSHOT
        Show details about a snapshot
    volume help
        Prints the nova manual
```

14.9 Status

```
Usage:
    status
    status db
    status CLOUD...

Shows system status

Description:
    status
        shows the status of al relevant subystems

    status db
        shows the status of the db

    status CLOUD...
        shows the status of the clouds specified
```

14.10 Stack (Hyungro)

```
OpenStack Heat DevOps Tools

Usage:
    stack start NAME [--template=TEMPLATE] [--param=PARAM]
    stack stop NAME
```

```
    stack show NAME
    stack list [--refresh] [--column=COLUMN] [--format=FORMAT]
    stack help | -h

Arguments:

  NAME           stack name
  help           Prints this message

Options:

   -v       verbose mode

Description:

      OpenStack Cloud supports software deployment with the Heat
      DevOps tool. Resources e.g. Security Group, Nova Server, or
      Floating IP can be defined to start a new stack.  Particular
      tasks defined in 'user_data' section will be executed by
      CloudInit on boot.

Examples:

      Start a Hadoop stack:
      cm> stack start hadoop --tempate=https://github.com/cloudmesh/cloudmesh/blob/master/heat-templa

      Stop a stack:
      cm> stack stop hadoop

      Show stack information:
      cm> stack show hadoop

      List running stacks:
      cm> stack list
```

14.11 notebook (not)

```
Usage:
    notebook create
    notebook start
    notebook delete

Manages the ipython notebook server

Options:

   -v       verbose mode
```

14.12 Project

```
Usage:
    project
    project info [--format=FORMAT]
    project default NAME
```

```
    project active NAME
    project delete NAME
    project completed NAME

Arguments:

    NAME           The project id
    FORMAT         The display format. (json, table)

Description:
    Manages the user's projects

    project info
        show project information
    project default
        set the default project
    project active
        set/add an active project,
    project delete
        delete the project
    project completed
        set a completed project, this will remove the project
        from active projects list and default project if it is
```

14.13 Loglevel

```
Usage:
    loglevel
    loglevel critical
    loglevel error
    loglevel warning
    loglevel info
    loglevel debug

    Shows current log level or changes it.

    loglevel - shows current log level
    critical - shows log message in critical level
    error    - shows log message in error level including critical
    warning  - shows log message in warning level including error
    info     - shows log message in info level including warning
    debug    - shows log message in debug level including info
```

14.14 Launcher

```
Software Launcher in Cloudmesh
- works with Chef Cookbooks and OpenStack Heat

Usage:
      launcher start MENU
      launcher stop STACK_NAME
      launcher list
      launcher show STACK_NAME
```

```
      launcher menu [--column=COLUMN] [--format=FORMAT]
      launcher import [FILEPATH] [--force]
      launcher export FILEPATH
      launcher help | -h

Arguments:

    MENU           Name of a cookbook
    STACK_NAME     Name of a launcher
    FILEPATH       Filepath
    COLUMN         column name to display
    FORMAT         display format (json, table)
    help           Prints this message

Options:

     -v       verbose mode

  Description:

      `launcher` command helps you to deploy software stacks on the
      cloud with Chef Cookbooks.  If you define your launcher sub
      command (menu) in ~/.cloudmesh/cloudmesh_launhcer.yaml,
      `launcher` command reads the YAML file and provides available
      software stacks.

Examples:

      Launcher start:
      cm> launcher start openmpi

      Launcher stop:
      cm> launcher stop openmpi

      List running launcher stacks:
      cm> launcher list

      List available launcher software stacks:
      cm> launcher menu
```

14.15 Key

```
Usage:
          key -h|--help
          key list [--source=SOURCE] [--dir=DIR] [--format=FORMAT]
          key add [--keyname=KEYNAME] FILENAME
          key default [KEYNAME]
          key delete KEYNAME

   Manages the keys



Arguments:

     SOURCE         mongo, yaml, ssh
```

```
     KEYNAME        The name of a key
     FORMAT         The format of the output (table, json, yaml)
     FILENAME       The filename with full path in which the key
                    is located

Options:

      --dir=DIR            the directory with keys [default: ~/.ssh]
      --format=FORMAT      the format of the output [default: table]
      --source=SOURCE      the source for the keys [default: mongo]
      --keyname=KEYNAME    the name of the keys

Description:


   key list --source=ssh  [--dir=DIR] [--format=FORMAT]

      lists all keys in the directory. If the directory is not
      specified the default will be ~/.ssh

   key list --source=yaml  [--dir=DIR] [--format=FORMAT]

      lists all keys in cloudmesh.yaml file in the specified directory.
       dir is by default ~/.cloudmesh

   key list [--format=FORMAT]

       list the keys in mongo

   key add [--keyname=keyname] FILENAME

       adds the key specifid by the filename to mongodb


   key list

        Prints list of keys. NAME of the key can be specified

   key default [NAME]

        Used to set a key from the key-list as the default key if NAME
        is given. Otherwise print the current default key

   key delete NAME

        deletes a key. In yaml mode it can delete only key that
        are not saved in mongo
```

14.16 Inventory (not, Gregor)

```
Usage:
       inventory clean
       inventory create image DESCRIPTION
       inventory create server [dynamic] DESCRIPTION
       inventory create service [dynamic] DESCRIPTION
       inventory exists server NAME
```

```
       inventory exists service NAME
       inventory
       inventory print
       inventory info [--cluster=CLUSTER] [--server=SERVER]
       inventory list [--cluster=CLUSTER] [--server=SERVER]
       inventory server NAME
       inventory service NAME

Manages the inventory

    clean       cleans the inventory
    server      define servers

Arguments:

  DESCRIPTION    The hostlist"i[009-011],i[001-002]"

  NAME           The name of a service or server


Options:

   v       verbose mode
```

14.17 Experiment (do, Gregor)

```
Usage:
       exp NOTIMPLEMENTED clean
       exp NOTIMPLEMENTED delete NAME
       exp NOTIMPLEMENTED create [NAME]
       exp NOTIMPLEMENTED info [NAME]
       exp NOTIMPLEMENTED cloud NAME
       exp NOTIMPLEMENTED image NAME
       exp NOTIMPLEMENTED flavour NAME
       exp NOTIMPLEMENTED index NAME
       exp NOTIMPLEMENTED count N

Manages the vm

Arguments:

  NAME           The name of a service or server
  N              The number of VMs to be started


Options:

   -v       verbose mode
```

14.18 debug (not cmd3, Gregor)

```
Usage:
      debug on
      debug off
```

```
    Turns the debug log level on and off.
```

14.19 Cluster

```
Create a Virtual Cluster

  Usage:
      cluster list [--format=FORMAT]
      cluster create NAME
                     [--count=COUNT]
                     [--user=USER]
                     [--cloud=CLOUD]
                     [--image=IMG|--imageid=IMGID]
                     [--flavor=FLAVOR|--flavorid=FLAVORID]
                     [--force]
      cluster show NAME
                   [--format=FORMAT]
                   [--column=COLUMN]
                   [--detail]
      cluster remove NAME
                     [--grouponly]


  Arguments:
      NAME        cluster name or group name

  Options:
      --count=COUNT              give the number of VMs to add into the cluster
      --user=USER                give the username
      --cloud=CLOUD              give a cloud to work on
      --flavor=FLAVOR            give the name of the flavor
      --flavorid=FLAVORID        give the id of the flavor
      --image=IMG                give the name of the image
      --imageid=IMGID            give the id of the image
      --force                    if a group exists and there are VMs in it, the program will
                                 ask user to proceed or not, use this flag to respond yes as
                                 default(if there are VMs in the group before creating this
                                 cluster, the program will include the exist VMs into the cluster)
      --grouponly                remove the group only without removing the VMs, otherwise
                                 cluster remove command will remove all the VMs of this cluster
      FORMAT                     output format: table, json, csv
      COLUMN                     customize what information to display, for example:
                                 --column=status,addresses prints the columns status
                                 and addresses
      --detail                   for table print format, a brief version
                                 is used as default, use this flag to print
                                 detailed table

  Description:
      Cluster Management

      cluster list
          list the clusters

      cluster create NAME --count=COUNT --user=USER [options...]
```

```
           Start a cluster of VMs, and each of them can log into all others.
           CAUTION: you should do some default setting before using
           this command:
           1. select cloud to work on, e.g. cloud select india
           2. activate the cloud, e.g. cloud on india
           3. set the default key to start VMs, e.g. key default [NAME]
           4. set the start name of VMs, which is prefix and index, e.g. label --prefix=test --id=1
           5. set image of VMs, e.g. default image
           6. set flavor of VMs, e.g. default flavor
           Also, it is better to choose a unused group name

       cluster show NAME
           show the detailed information about the cluster VMs

       cluster remove NAME [--grouponly]
           remove the cluster and its VMs, if you want to remove the cluster(group name)
           without removing the VMs, use --grouponly flag

   Examples:

         Create Virtual Cluster consist of 3 VM instances with m1.small
         server size and Ubuntu 14.04 base image
         cm> cluster create vc --count=3  --image=futuresystems/ubuntu-14.04 --flavor=m1.small
```

14.20 Admin

```
Usage:
  admin password reset
  admin version

Options:


Description:
    admin password reset
       reset portal password

    admin version
       prints the version numbers of cloudmesh and its plugins
```

#.. toctree:: # :caption: outdated # :maxdepth: 4

choco # cloudmesh_base # cloudmesh_client # commands_nova # commands_register # docker # man # old1 # system-cygwin

CHAPTER 15

Indices and tables

- genindex
- modindex
- search