DevStack: <http://devstack.org/>

1. Environment

|  |
| --- |
| For the purpose of this tutorial, we will be running DevStack.  - Ubuntu 12.04  - At least 2 GB of RAM  For IDE, use any of your favorite Python IDE or text editor:  - PyCharm  - PyDev  - Komodo  - Emacs  - Vi  If you are a developer, I recommend getting a VM since DevStack installs a lot of dependencies, and makes a lot of changes to your system.  There are a lot of options for getting a VM:  1. Using Virtualbox  2. Using VMWare Fusion  3. ICS Joyent FlexCloud  4. ICS OpenStack Cloud  Also checkout Vagrant as it now supports all of the above. |

**Install devstack on ICS Joyent FlexCloud**

|  |
| --- |
| From Joyent customer portal (10.245.122.6) create a new virtual machine with 4GB RAM profile:  > Login into your Joyent account  > Click on "order a machine"  > Scroll down and choose the "Machine Image" of type "ubuntu-12.04-openstack"  > Choose the profi e2GB B RAM named "regular\_4096" (name is visible when you mouse over the 4GB profile)  > Choose ICS as the Datacenter  > You can give the VM a name or leave the field empty and a default ID will be given.  > Click on Provision.  > After VM is running ssh into it using the IP address.  > Before compiling devStack you will need to add an entry for your hostname to the /etc/hosts file.  echo "<VM IP ADDRESS> $HOSTNAME" >> /etc/hosts  > The above command will append the IP address mapping to hostname entry into the /etc/hosts file.  > Now follow the instructions for devStack building specified at:  2.) |

2. Install

|  |
| --- |
| Create a Ubuntu 12.04 (Precise) server, and copy the ip address of the server.  ssh root@$server\_ip  adduser stack  echo "stack ALL=(ALL) NOPASSWD: ALL" >> /etc/sudoers  apt-get update; apt-get install -qqy git  exit  ssh stack@$server\_ip  git clone https://github.com/openstack-dev/devstack.git  cd devstack  cat >> localrc <<EOF  DEST=/opt/stack  ADMIN\_PASSWORD=admin  MYSQL\_PASSWORD=admin  RABBIT\_PASSWORD=admin  SERVICE\_TOKEN=admin  SERVICE\_PASSWORD=admin  LOGFILE=/opt/stack/logs/stack.log  SCREEN\_LOGDIR=/opt/stack/logs  VERBOSE=True  ## Controller Host ##  # HOST\_IP=<IP ADDRESS>  # MULTI\_HOST=1  ## Network nova-network ##  FLAT\_INTERFACE=eth0  FIXED\_RANGE=172.24.17.0/24  FIXED\_NETWORK\_SIZE=254  FLOATING\_RANGE=192.168.1.128/25  ## Leaving Default Services Enabled ##  DISABLED\_SERVICES=quantum  ## Logs ##  # LOGFILE=/opt/stack/logs/stack.sh.log  # VERBOSE=True  # LOG\_COLOR=False  # SCREEN\_LOGDIR=/opt/stack/logs  ENABLED\_SERVICES+=,swift  SWIFT\_HASH=66a3d6b56c1f479c8b4e70ab5c2000f5  SWIFT\_REPLICAS=1  SWIFT\_DATA\_DIR=$DEST/data  #  SCREEN\_LOGDIR=$DEST/logs/screen  EOF  export PIP\_DEFAULT\_TIMEOUT=120  ./stack.sh  - Ensure that it passes successfully  - Now run the following command to change some kernel parms  sed -i "s/guest.acpi = True/guest.acpi = False/" /opt/stack/nova/nova/virt/libvirt/driver.py  sed -i "s/guest.apic = True/guest.apic = False/" /opt/stack/nova/nova/virt/libvirt/driver.py  - Now change the network range to not conflict with ICS network:  vi /etc/nova/nova.conf  - Set the below field to the specified value.  fixed\_range = 172.24.17.0/24  - Now restart nova-compute by:  screen –r stack  ctrl+a n <- This will take you to the next screen – keep pressing "n" until  - At the bottom of the screen you will see a "\*" sign next to a name, when the "\*" is against the n-cpu name.  ctrl+c <- To stop the service  - Use up arrow to chose the previous command and press ENTER  ctrl+a d <- to Detach from the screen  Now Follow the instructions at 3.) |
|  |

3.

|  |
| --- |
|  |

cd /home/stack/devstack

source openrc

nova --help

nova list

nova image-list

nova flavor-list

nova boot --image cirros-0.3.1-x86\_64-uec --flavor m1.tiny MyFirstInstance

nova list

ipaddress=... # Get IP address from nova list

ping $ipaddress

nova show MyFirstInstance

ssh cirros@$ipaddress

# password is cubswin:)

uname -a

free -m

ifconfig

ping googe.com

exit

nova --debug list

nova console-log MyFirstInstance

nova delete MyFirstInstance

4.

|  |
| --- |
|  |

nova-manage --help

nova-manage service list

nova-manage vm list

nova-manage host list

nova-manage fixed list

5.

|  |
| --- |
|  |

nova keypair-add test > test.pem

chmod 600 test.pem

nova keypair-list

nova boot --image cirros-0.3.1-x86\_64-uec --flavor m1.tiny --key-name test MySecondInstance

nova list

> /home/stack/.ssh/known\_hosts

ipaddress=... # Get IP address from nova list

ssh -i test.pem cirros@$ipaddress

cat /home/cirros/.ssh/authorized\_keys

exit

nova delete MySecondInstance

6.

|  |
| --- |
|  |

glance image-list

IMAGE="cirros-0.3.0-x86\_64-disk.img"

wget -nc http://launchpad.net/cirros/trunk/0.3.0/+download/$IMAGE

glance image-create --name "test-image" --public --container-format bare --disk-format qcow2 < "$IMAGE"

glance image-list

nova boot --image test-image --flavor m1.tiny --key\_name test MyThirdInstance

nova list

nova delete MyThirdInstance

7.

|  |
| --- |
|  |

# Create the user-data script

cat > ~/user-data.txt <<EOF

#!/bin/sh

echo "Hello World. The time is now $(date -R)!" | tee /home/cirros//output.txt

EOF

# Create a new instance with the user-data

nova boot --image cirros-0.3.1-x86\_64-uec --flavor m1.tiny --key\_name test --user\_data ~/user-data.txt MyUserDataInstance

nova list

> /home/stack/.ssh/known\_hosts

ipaddress=... # Get IP address from nova list

ssh -i test.pem cirros@$ipaddress

cat output.txt

curl http://169.254.169.254/

curl http://169.254.169.254/latest

curl http://169.254.169.254/latest/user-data

exit

# Nova-network configures iptables to NAT port 80 of 169.254.169.254 to the $my-ip

sudo iptables -L -n -t nat | grep 169

# my\_ip address is specified in /etc/nova/nova.conf

cat /etc/nova/nova.conf | grep my\_ip

# meta-data api is running on the api server

cat /etc/nova/nova.conf | grep enabled\_apis

nova delete MyUserDataInstance

8.

|  |
| --- |
|  |

http://server\_ip

9.

|  |
| --- |
|  |

cat /etc/nova/nova.conf

What is the virtualization type in use?

What type of scheduler is used?

What network model is used?

What is the debug level?

Where are the instances stored?

What is the messaging driver in use?

10.

|  |
| --- |
|  |

cd /opt/stack/logs

tail -f screen-n-api.log

11.

|  |
| --- |
|  |

cd /opt/stack/nova

./run\_tests.sh -h

./run\_tests.sh -p

./run\_tests.sh scheduler

./run\_tests.sh test\_libvirt:HostStateTestCase

12.

|  |
| --- |
|  |

# List screens

screen -ls

# Attach to a screen

screen -r stack

# Next screen

ctrl+a n

# Previous screen

ctrl+a p

# Stop a service

ctrl+c

# Restart the service

Use up arrow to choose the previous command and press ENTER

# Detach from screen

ctrl+a d

nova list

13.

|  |
| --- |
|  |

nova boot --image test-image --flavor m1.tiny MyVirshInstance

nova list

virsh --help

sudo virsh list --all

# Running vm instances

cd /opt/stack/data/nova/instances

# Used for local caching of vm inages

cd /opt/stack/data/nova/instances/\_base

nova delete MyVirshInstance

14.

|  |
| --- |
|  |

cat /etc/nova/nova.conf | grep mysql

mysql -u root -padmin

show databases;

use nova;

show tables;

select \* from instances\G;

select \* from compute\_nodes\G;

exit

15.

|  |
| --- |
|  |

sudo rabbitmqctl

sudo rabbitmqctl status

sudo rabbitmqctl list\_queues

sudo rabbitmqctl list\_exchanges

sudo rabbitmqctl list\_bindings

16.

|  |
| --- |
|  |

cd /opt/stack/nova

|-bin # Starter scripts for all nova services

|

|-doc # Documentation

|

|-etc # Configuration files

|---nova

|-----rootwrap.d

|

|-nova

|---api # ReST API

|-----ec2

|-----metadata # Provides instance specific data to the instance

|-----openstack

|-------compute # APIs

|---------contrib # Extensions

|

|---cells # Scaling and geographical distribution of nova clusters

|

|---compute # Handles all processes related to vms

|

|---conductor

|

|---db # Database interface for all services

|-----sqlalchemy

|-------migrate\_repo

|

|---image # Storage backend for images

|

|---locale # Translations of errors and messages

|

|---network # Network service or interface to Quantum

|-----quantumv2

|

|---openstack # Reusable code between OpenStack projets

|-----common

|

|---scheduler # Scheduler service to choose the appropriate host

|-----filters

|-----weights

|

|---tests # Unit tests

|

|---virt # Virt drivers

|-----baremetal # Exposes hardware via Nova API, uses PXE and IPMI

|-----hyperv

|-----libvirt # qemu, kvm, lxc

|-----powervm

|-----vmwareapi

|-----xenapi

|

|---volume # Block Storage service or interface to Cinder

17.

|  |
| --- |
|  |

# Download a sample api controller

curl -L https://raw.github.com/sacharya/openstack-101/master/utsaservers.py > /opt/stack/nova/nova/api/openstack/compute/contrib/utsaservers.py

screen -r stack

# Next screen until you reach n-api service

ctrl+a n

# Stop the n-api service.

ctrl+c

# Restart the n-api service

Use up arrow to choose the previous command and press ENTER

# Detach from screen

ctrl+a d

nova --debug list

# Grab the curl command from the debug output

nova --debug list 2>&1 | tee my.log | grep 'curl.\*detail' | cut -c6- | tee utsa\_servers.sh

sed -i "s/servers\\/detail/os-utsaservers/g" utsa\_servers.sh

chmod +x utsa\_servers.sh

# Make a call to the sample api called os-utsaservers

./utsa\_servers.sh

18.

|  |
| --- |
| **from** nova **import** compute  **from** nova.api.openstack **import** extensions  **class** **UtsaserverController**(object):  """the Utsaservers API controller """  **def** **\_\_init\_\_**(self):  self**.**compute\_api **=** compute**.**API()  **def** **index**(self,req, deleted**=**True):  """return a list of servers """  context **=** req**.**environ['nova.context']  search\_opts **=** {'deleted': deleted}  instances **=** self**.**compute\_api**.**get\_all(context, search\_opts**=**search\_opts)  servers **=** []  **for** instance **in** instances:  server **=** {'id': instance**.**get('uuid'),  'local\_id': instance**.**get('id'),  'name': instance**.**get('display\_name'),  'status': instance**.**get('vm\_state'),  'host': instance**.**get('host'),  'deleted': instance**.**get('deleted'),  'deleted\_at':instance**.**get('deleted\_at'),  'tenant\_id': instance**.**get('project\_id')}  servers**.**append(server)  **return** {'servers': servers}  **def** **create**(self,req, body):  """create a server """  **pass**  **def** **show**(self,req, id):  """ read the details of a server given its id"""  **pass**  **def** **update**(self, req, id, body):  """updates a server given its id and content"""  **pass**  **def** **delete**(self,req, id):  """removes a server given its id"""  **pass**  **class** **Utsaservers**(extensions**.**ExtensionDescriptor):  """ExtensionDescriptor implementation"""  name **=** "Utsaservers"  alias **=**"os-utsaservers"  namespace **=** "http://docs.openstack.org/compute/ext/utsaservers/api/v1.1"  updated **=** "2011-12-23T00:00:00+00:00"  **def** **get\_resources**(self):  """ register the new Utsaservers RESTful resource """  resources **=** [extensions**.**ResourceExtension('os-utsaservers',UtsaserverController())]  **return** resources |

**from** nova **import** compute

**from** nova.api.openstack **import** extensions

**class** **UtsaserverController**(object):

"""the Utsaservers API controller """

**def** **\_\_init\_\_**(self):

self**.**compute\_api **=** compute**.**API()

**def** **index**(self,req, deleted**=**True):

"""return a list of servers """

context **=** req**.**environ['nova.context']

search\_opts **=** {'deleted': deleted}

instances **=** self**.**compute\_api**.**get\_all(context, search\_opts**=**search\_opts)

servers **=** []

**for** instance **in** instances:

server **=** {'id': instance**.**get('uuid'),

'local\_id': instance**.**get('id'),

'name': instance**.**get('display\_name'),

'status': instance**.**get('vm\_state'),

'host': instance**.**get('host'),

'deleted': instance**.**get('deleted'),

'deleted\_at':instance**.**get('deleted\_at'),

'tenant\_id': instance**.**get('project\_id')}

servers**.**append(server)

**return** {'servers': servers}

**def** **create**(self,req, body):

"""create a server """

**pass**

**def** **show**(self,req, id):

""" read the details of a server given its id"""

**pass**

**def** **update**(self, req, id, body):

"""updates a server given its id and content"""

**pass**

**def** **delete**(self,req, id):

"""removes a server given its id"""

**pass**

**class** **Utsaservers**(extensions**.**ExtensionDescriptor):

"""ExtensionDescriptor implementation"""

name **=** "Utsaservers"

alias **=**"os-utsaservers"

namespace **=** "http://docs.openstack.org/compute/ext/utsaservers/api/v1.1"

updated **=** "2011-12-23T00:00:00+00:00"

**def** **get\_resources**(self):

""" register the new Utsaservers RESTful resource """

resources **=** [extensions**.**ResourceExtension('os-utsaservers',UtsaserverController())]

**return** resources