Table of Contents

1. INTRODUCTION TO OPENSTACK ................................................................. 9
   1.1 ORIGINS OF OPENSTACK ................................................................. 9
   1.2 ROLE OF THE OPENSTACK FOUNDATION ..................................... 9
   1.3 OPENSTACK SERVICES ................................................................. 10
       1.3.1 Nova ‘Compute’ Service ....................................................... 10
       1.3.1 Neutron ‘Networking’ Service ........................................... 11
       1.3.2 Swift ‘Object Storage’ service .......................................... 12
       1.3.3 Cinder ‘Block Storage’ service ......................................... 13
       1.3.4 Keystone ‘Identity’ service ............................................... 13
       1.3.5 Glance ‘Image store’ service ........................................... 14
       1.3.6 Other Services ................................................................. 14
   1.4 BEHIND THE CORE OPENSTACK PROJECTS .................................. 14
       1.4.1 The RESTful API ............................................................... 15
   1.5 OPENSTACK RELEASES ................................................................. 15

2. OPENSTACK TRAINING LABORATORY ...................................................... 17
   2.1 ARCHITECTURE ................................................................................. 17
   2.2 CONTROLLER NODE ......................................................................... 18
   2.3 COMPUTE NODE ............................................................................. 18
       2.3.1 Networking ........................................................................... 18
   2.4 PASSWORDS ..................................................................................... 19

3. OPENSTACK TRAINING LABS PRE-INSTALLATION .................................... 21
   3.1 GET GIT ............................................................................................... 21
   3.2 CLONE THE TRAINING LABS ........................................................... 21
   3.3 UPGRADE THE TRAINING LABS ..................................................... 21
   3.4 CLUSTER TRAINING DIRECTORY VARIABLES ................................ 21
   3.5 PRE-INSTALLATION CHECK ............................................................. 22
       3.5.1 Enable virtualisation support in BIOS .................................. 22
   3.6 OPTIMISE THE NODES ..................................................................... 23
   3.7 ENABLE HEAT SERVICE ................................................................... 24
   3.8 LOG FILES ......................................................................................... 24
   3.9 ADD CONTROLLER AND COMPUTE1 IP TO HYPERVISOR HOSTS FILE .. 25

4. SETUP OPENSTACK TRAINING LABS ON KVM/QEMU .......................... 27
   4.1 INSTALLATION .................................................................................. 28
       4.1.1 Install KVM packages ............................................................ 28
   4.2 GNU/LINUX BRIDGE UTILITIES .................................................... 28
   4.3 VIRT-MANAGER ................................................................................ 28
   4.4 BUILD INTRODUCTION ................................................................... 30
   4.5 BUILD STEPS ..................................................................................... 31
   4.6 RUN THE STACKTRAIN SCRIPT ..................................................... 32
       4.6.1 Stacktrain ................................................................................. 32
       4.6.2 Confirm installed release ...................................................... 32
       4.6.3 Memory and harddisks ......................................................... 32
   4.7 USING THE CLUSTER ....................................................................... 33
       4.7.1 Review the running VMs ......................................................... 33
4.7.2 Controller node........................................................................................................... 33
4.7.3 Compute node.............................................................................................................. 34
4.7.4 VM IP addresses......................................................................................................... 34
4.8 REVIEWING THE NETWORKS CREATED BY THE SCRIPT............................................. 35
4.9 ACCESS THE CONTROLLER NODE.................................................................................. 36
4.10 ACCESS THE COMPUTE NODE....................................................................................... 36
4.11 ADD HYPERVERSOR SSH KEYS TO THE CONTROLLER AND COMPUTE1 NODES........ 37

5. SETUP OPENSTACK TRAINING LABS ON VIRTUALBOX............................................. 39
   5.1 BUILD INTRODUCTION.................................................................................................. 40
   5.2 BUILD STEPS............................................................................................................... 40
   5.3 RUN THE SCRIPTS........................................................................................................ 41
      5.3.1 Stacktrain.............................................................................................................. 41
      5.3.2 Confirm installed release....................................................................................... 41
      5.3.3 Memory and harddisks......................................................................................... 41
   5.4 USING THE CLUSTER.................................................................................................... 42
      5.4.1 Review the running VMs..................................................................................... 42
      5.4.2 Controller node................................................................................................... 43
      5.4.3 Compute node....................................................................................................... 46
      5.4.4 VM IP addresses................................................................................................. 48
   5.5 REVIEWING THE NETWORKS CREATED BY THE SCRIPT............................................ 49
   5.6 ACCESS THE CONTROLLER NODE.................................................................................. 49
   5.7 ACCESS THE COMPUTE NODE....................................................................................... 50
   5.8 ADD HYPERVERSOR SSH KEYS TO THE CONTROLLER AND COMPUTE1 NODES........ 50

6. OPERATING THE OPENSTACK TRAINING TESTBED ON KVM/QEMU................. 53
   6.1 MANAGING KVM/QEMU VM IN HEADLESS MODE................................................... 53
   6.2 STARTING THE VMs..................................................................................................... 53
   6.3 POWERING OFF THE VMs........................................................................................... 53
   6.4 SAVING VM STATE AND STOPPING.......................................................................... 54
   6.5 MANAGING SNAPSHOTs.............................................................................................. 54
   6.6 TAKING A SNAPSHOT.................................................................................................. 54
   6.7 RESTORING TO A PREVIOUS SNAPSHOT................................................................... 54
   6.8 DELETE A SNAPSHOT.................................................................................................. 55
   6.9 INCREASE THE SIZE OF THE COMPUTE1 NODE.......................................................... 55

7. OPERATING THE OPENSTACK TRAINING TESTBED ON VIRTUALBOX.................. 59
   7.1 MANAGING VIRTUALBox VM IN HEADLESS MODE..................................................... 59
   7.2 STARTING THE VMs..................................................................................................... 59
   7.3 POWERING OFF THE VMs........................................................................................... 59
   7.4 SAVING VM STATE AND STOPPING.......................................................................... 59
   7.5 MANAGING SNAPSHOTs.............................................................................................. 60
   7.6 TAKING A SNAPSHOT.................................................................................................. 60
   7.7 RESTORING TO A PREVIOUS SNAPSHOT................................................................... 60
      DELETE A SNAPSHOT.................................................................................................... 60
   7.8 INCREASE THE SIZE OF THE COMPUTE1 NODE.......................................................... 61

8. REVIEWING THE INSTALLATION...................................................................................... 67
   8.1 CONTROLLER NODE - DATABASE.............................................................................. 67
   8.2 CLIENT ENVIRONMENT SCRIPTS................................................................................ 70
   8.3 IDENTITY SERVICE - KEYSTONE.................................................................................. 71
      8.3.1 Controller node...................................................................................................... 71
8.4 IMAGE SERVICE - GLANCE................................................................. 73
8.4.1 Controller node........................................................................... 73
8.5 COMPUTE SERVICE - NOVA............................................................ 74
8.5.1 API............................................................................................... 74
8.5.2 Compute core................................................................................. 74
8.5.3 Networking for VMs................................................................. 74
8.5.4 Console interface........................................................................ 75
8.5.5 Image management................................................................. 75
8.5.6 Command-line clients and other interfaces............................... 75
8.5.7 Other components................................................................. 75
8.5.8 Controller node........................................................................... 76
8.6 NETWORKING SERVICE - NEUTRON................................................. 76
8.6.1 Controller node........................................................................... 77
8.6.2 Networking.................................................................................. 79
8.6.3 Masquerade on virtualBox host............................................... 80
8.7 BLOCK STORAGE SERVICE - CINDER............................................... 80
8.7.1 Controller node........................................................................... 81
8.8 ORCHESTRATION SERVICE - HEAT.................................................. 81
8.9 INSTANCE FLAVOUR........................................................................ 83

9. DEPLOYING A VM INSTANCE................................................................ 85
9.1 DEPLOYING AN INSTANCE.............................................................. 85
9.2 CONFIGURE SDN............................................................................ 86
9.3 CONTROLLER NODE........................................................................ 87
9.3.1 Generate a key pair..................................................................... 87
9.3.2 Security group............................................................................ 89
9.3.3 Create volume........................................................................... 92
9.3.4 Launch a Cirros instance........................................................... 94
9.3.5 Attach the volume..................................................................... 97
9.3.6 Connect to the new instance....................................................... 98
9.3.7 Provider network gateway......................................................... 99
9.3.8 Host public interface............................................................... 100
9.3.9 IP Address on the public Internet............................................. 100
9.4 REVIEW THE INSTANCE................................................................... 100
9.4.1 Controller node........................................................................... 100
9.4.2 Compute node............................................................................ 101
9.5 CONFIGURE A VOLUME IN THE NEW INSTANCE............................. 103

10. SCRIPTING THE BUILDING AND LAUNCHING NEW INSTANCE............... 105
10.1 CLEANING THE NODES.................................................................. 105
10.2 LAUNCH NEW INSTANCE............................................................... 105
10.2.1 Confirm VM instance........................................................... 108

11. DOWNLOAD AND BUILD A SECOND UBUNTU IMAGE............................ 109
11.1 COMPUTE NODE - KVM/QEMU ................................................... 109
11.2 COMPUTE NODE - VIRTUALBOX............................................... 111
11.3 GLANCE - IDENTITY SERVICE.................................................. 112
11.4 FLAVOUR................................................................................... 114

12. ADDING AN ADDITIONAL COMPUTE NODE ON KVM/QEMU................. 117
12.1 CLONE THE COMPUTE VM.......................................................... 117
12.1.1 Start clone.............................................................................. 117
12.1.2 Connect to clone..................................................................... 118
12.1.3 Reconfigure clone............................................................... 118
12.1.4 Reboot instance and login to see changes.................................................................119
12.2 START THE CONTROLLER AND COMPUTE1..............................................................119
12.2.1 Adjust host table in the compute1........................................................................120
12.2.2 Adjust the host table of the controller.................................................................121
12.2.3 Configure compute2..............................................................................................122
12.3 CHECK THE CONTROLLER FOR COMPUTE2.............................................................123

13. ADDING AN ADDITIONAL COMPUTE NODE ON VIRTUALBOX.................................125
13.1 CLONE THE COMPUTE VM.......................................................................................125
13.1.1 Start clone..............................................................................................................125
13.1.2 Connect to clone......................................................................................................125
13.1.3 Reconfigure clone..................................................................................................126
13.1.4 Reboot instance and login to see changes.............................................................127
13.2 START THE CONTROLLER AND COMPUTE1..............................................................127
13.2.1 Adjust host table in the compute1........................................................................128
13.2.2 Adjust the host table of the controller.................................................................128
13.2.3 Configure compute2..............................................................................................129
13.3 CHECK THE CONTROLLER FOR COMPUTE2.............................................................129

14. KVM/QEMU RESTARTING PROCEDURE AFTER SHUTDOWN....................................131
14.1 LIST AND ENABLE NETWORKS.................................................................................131
14.2 START THE NODES.....................................................................................................132

15. WORKING WITH THE HORIZON DASHBOARD............................................................133
15.1 ACCESSING HORIZON ON KVM/QEMU TESTBED......................................................133
15.2 ACCESSING HORIZON ON THE VIRTUAL BOX TESTBED..........................................134
15.3 LOGGING IN...............................................................................................................135
15.4 ‘ADMIN’ USER FUNCTIONS.......................................................................................136
15.4.1 Create a new project..............................................................................................136
15.4.2 Create a flavour......................................................................................................137
15.4.3 Create a new user....................................................................................................138
15.5 PROJECT USER FUNCTIONS.......................................................................................139
15.5.1 Create a Security Group.......................................................................................140
15.5.2 Create an instance.................................................................................................142
15.6 RUN THE nat_tables.sh SCRIPT.............................................................................145
15.7 CONNECT TO THE INSTANCE....................................................................................145

16. CREATING NETWORKS...............................................................................................147
16.1 INITIAL CONFIGURATION..........................................................................................148
16.2 CREATE PRIVATE NETWORK......................................................................................148
16.3 CREATE HOSTS ON THE PROVIDER AND PRIVATE NETWORKS...............................148
16.4 CREATE A ROUTER....................................................................................................148
16.5 ADD A ROUTE ON THE HYPERVISOR TO THE PRIVATE NETWORK.........................149
16.6 TEST THE CONFIGURATION.....................................................................................149
16.7 REVIEW TOPOLOGY ON THE HORIZON DASHBOARD..............................................150
16.8 SCRIPTING THE OPERATION.....................................................................................151

17. USING HEAT ORCHESTRATION...................................................................................157
17.1 INTRODUCTION..........................................................................................................157
17.2 HEAT ORCHESTRATION TEMPLATES (HOT)..............................................................158
17.2.1 Template version....................................................................................................158
17.2.2 Description..............................................................................................................158
17.2.3 Parameter groups...................................................................................................158
17.2.4 Parameters........................................................................................................................................159
17.2.5 Resources........................................................................................................................................159
17.2.6 Outputs............................................................................................................................................159
17.2.7 Conditions.......................................................................................................................................159
17.3 CREATING SINGLE SERVERS........................................................................................................160
17.4 CREATE COMPLETE NETWORK AND SERVERS..........................................................................166
   17.4.1 Networks – child template.............................................................................................................166
   17.4.2 Delete stack....................................................................................................................................170
   17.4.3 Parent template.............................................................................................................................171
   17.4.4 Stack events....................................................................................................................................174
   17.4.5 Add a route on the hypervisor to the private network.................................................................176
   17.4.6 Test the configuration.....................................................................................................................177
   17.4.7 Review topology on the Horizon dashboard..................................................................................178

18. APPENDICES.........................................................................................................................................179
   18.1 APPENDIX 1 - NAT MASQUERADE SCRIPT FOR HYPervisor HOST.............................................179
   18.2 APPENDIX 2 – CLUSTER START/STOP SCRIPT..............................................................................180
      18.2.1 Running for a KVM/QEMU system.............................................................................................182
      18.2.2 Running for a VirtualBox system...............................................................................................183
   18.3 APPENDIX 3 - CLEAN NODES SCRIPT FOR HYPervisor HOST....................................................184
      18.3.1 Running for a KVM/QEMU system.............................................................................................186
      18.3.2 Running for a VirtualBox system...............................................................................................187
   18.4 APPENDIX 4 - SCRIPT TO LAUNCH A VM INSTANCE....................................................................188
   18.5 APPENDIX 5 - SCRIPT TO LAUNCH A NETWORK WITH VMs.......................................................190
   18.6 APPENDIX 6 - STACKTRAIN CLUSTER CREATION SCRIPT – KVM.............................................192
   18.7 APPENDIX 7 - STACKTRAIN CLUSTER CREATION SCRIPT – VirtualBox....................................196

19. ABBREVIATIONS...................................................................................................................................201

20. BIBLIOGRAPHY....................................................................................................................................203
Illustration Index

Illustration 1: OpenStack - Projects navigator.................................................................10
Illustration 2: Neutron networking..................................................................................11
Illustration 3: Swift 'Object Storage' service......................................................................12
Illustration 4: OpenStack Laboratory architecture..........................................................17
Illustration 5: KVM virtualisation block diagram..............................................................27
Illustration 6: virt-manager via SSH..................................................................................29
Illustration 7: Deploy an instance.......................................................................................83
Illustration 8: Floating IP addresses...................................................................................84
Illustration 9: Virtual Console............................................................................................94
Illustration 10: Ubuntu instance........................................................................................111
Illustration 11: Horizon login - KVM/QEMU testbed........................................................129
Illustration 12: Horizon login - VirtualBox testbed............................................................130
Illustration 13: Admin opening dashboard screen...............................................................131
Illustration 14: Create Project...........................................................................................132
Illustration 15: Create Flavour..........................................................................................133
Illustration 16: Create User...............................................................................................134
Illustration 17: Project User opening dashboard screen....................................................135
Illustration 18: Create Security Group................................................................................136
Illustration 19: Adding rules.............................................................................................136
Illustration 20: Launch Instance - Details..........................................................................137
Illustration 21: Instance Launch - Source..........................................................................137
Illustration 22: Launch Instance - Flavour.........................................................................138
Illustration 23: Add the Security Group.............................................................................138
Illustration 24: Instance launched......................................................................................139
Illustration 25: Simple network.........................................................................................141
Illustration 26: Network topology......................................................................................145
Illustration 27: Network graph..........................................................................................145
Illustration 28: HEAT functional diagram.........................................................................153
Illustration 29: Network topology......................................................................................166
Illustration 30: Network graph..........................................................................................166
Illustration 31: Network topology......................................................................................174
Illustration 32: Network graph..........................................................................................174
1. **Introduction to OpenStack**

Throughout the years, corporate computing has seen many developments, which have eventually lead to the rise of cloud computing as we know it today. In the 1990s, corporate computing was centred around servers in a data centre. In 2000s, corporate computing was largely based on virtualisation. In the 2010s, we have witnessed the rise of cloud computing to leverage corporate computing. The concept of *cloud computing* is very broad, to the point that we can affirm that cloud computing is a concept rather than a particular technological development. If you ask an end-user to explain what cloud computing is, and then ask a system administrator the same question, you will get two different descriptions. In general, there are three important approaches when it comes to cloud computing:

- **Infrastructure as a Service (IaaS):** an infrastructure that is used to provide Virtual Machines (VM)
- **Platform as a Service (PaaS):** the provider supplies the network, servers, storage, OS and middleware to host an application
- **Software as a Service (SaaS):** the provider gives access to an application

OpenStack belongs to the IaaS cloud computing category. However, OpenStack is continuously evolving, broadening its scope. On occasion, the focus of OpenStack goes beyond IaaS.

1.1 **Origins of OpenStack**

OpenStack started in 2010, as a joint project of Rackspace Hosting and National Aeronautics and Space Administration (NASA). NASA contributed their Nebula platform, which later developed into Nova. Rackspace contributed their Cloud Files platform, which later became Swift. In April of 2011, the OpenStack Bexar release was introduced in Ubuntu. Later that same year, Debian GNU/Linux included OpenStack Cactus in their distribution. In 2012, Red Hat announced a preview of their OpenStack distribution as well. Since then, many others followed, including Oracle, HP, and Vmware (now owned by Dell).

1.2 **Role of the OpenStack Foundation**

The OpenStack Foundation promotes the global development, distribution, and adoption of the cloud operating system. It provides shared resources to grow the OpenStack cloud. It also enables technology vendors and developers to assist in the production of cloud software. See: [https://www.openstack.org/foundation/](https://www.openstack.org/foundation/).
1.3 OpenStack Services

Illustration 1 shows a selection of the many OpenStack services. Each service webpage can be accessed on the Internet via:
https://www.openstack.org/software/project-navigator/

1.3.1 Nova 'Compute' Service

Nova is the most important core project in OpenStack. It handles the Compute environment, the VM instance lifecycle. Nova service is not a hypervisor in itself but interfaces to a number of different hypervisors like Xen, Kernel Virtual Machine (KVM)/Quick Emulator (QEMU), VMware, vSphere. Nova installs an agent on the hypervisor so it is supported on the OpenStack environment.

The Nova service is responsible for spawning, scheduling, and decommissioning of VMs on demand. It includes Nova service processes that are running on the cloud controller node, as well as Nova agents, that are running on the compute nodes.
1.3.1 Neutron ‘Networking’ Service

OpenStack Neutron enables Software Defined Networking (SDN). SDN allows users to define their own networking between the instances that are deployed. Illustration 2 demonstrates a typical OpenStack environment. In the environment there are a number of different compute nodes connected by using a physical underlay network involving routing functionality. The OpenStack user is not aware of the detail of the underlay. The user can see an abstraction network at a higher level, that is called the Overlay Network. SDN permits the User to create logical networks that do not require the consideration of the underlying physical network. In fact the User will most likely be unaware of the topology of the underlay network.

The Neutron service manages this by interfacing with the physical network architecture using a pluggable architecture that supports many networking vendors and technologies.

Furthermore the Neutron service also provides an API for users to define networks and the attachments into them.
1.3.2 Swift ‘Object Storage’ service

The Swift ‘Object Storage’ service provides scalability at the storage level. It works with binary objects to store data in a distributed, replicated way. Hard-drives are physical devices, they are limited and they are not very scalable.

The Swift service provides scalability by providing an object-based storage model. An application normally, in order to write data, writes to a file. In an OpenStack environment, the application writes to a file but not to a hard drive, the application via the Cinder ‘Block Storage’ service interfaces with Swift ‘Object Storage’ service over a RESTful API which is in turn can communicate with many, many storage nodes. Swift uses a proxy service which, when it receives data from Cinder, creates chunks of data called binary objects.

As demonstrated in Illustration 3 the received data is broken into three binary objects (a), (b), and (c). In Swift, binary objects (a) may be stored in the first storage node, and binary object (b) in the second storage node with binary object (c) stored in the third storage node. To create fault tolerance Swift includes a replication algorithm which stores the binary objects on multiple storage nodes. By default it does this three times but it is possible to do it more times if necessary.

Efficiency is also achieved because, the moment that the application needs to retrieve the data, it will address the Swift proxy via the Cinder service which uses an advanced algorithm to determine exactly where the binary objects reside. It then sends calls to all the storage nodes that are involved, these are capable of working in parallel. The data will arrive at the Swift proxy, and onwards to the application via Cinder quickly and efficiently.

Illustration 3: Swift ‘Object Storage’ service
If the storage nodes are for example one terabyte (TB) each and storage is running low, more Swift storage nodes can simply be added, and the binary objects rebalanced as set in the Swift storage configuration.

The Swift proxy is communicated with using a Restful API. REST is a standard way of communicating in an OpenStack environment. The application is not writing a file to a filesystem, it is using a RESTful API call, which is understood by the Swift proxy. This API permits the Create, Read, Update and Delete (CRUD) functions.

RESTful API is the native language of OpenStack, and that makes Swift the native choice for object storage in OpenStack.

An alternative to using Swift ‘Object Storage’ service is Ceph. Ceph is a similar distributed object store and file system designed to provide excellent performance, reliability and scalability.

1.3.3 Cinder ‘Block Storage’ service
Cinder block storage provides persistent storage to instances. By default, the storage in the VM instances is ephemeral, non-persistent. In other words the contents of the VM is lost when that VM is shut down.

Cinder allows administrators to attach additional persistent block devices to instances such that data can be saved.

The Cinder interface specifies a number of discrete functions, including basic functionality such as create volume, delete volume and attach. There are also more advanced functions that can extend volumes, take snapshots, clone volumes and create volumes from a VM image.

The Cinder service can use different back-ends, as well. It can be local storage like local Linux Logical Volume Manager (LVM), or it can include Swift and Ceph object storage as well for increased scalability.

1.3.4 Keystone ‘Identity’ service
The Keystone Identity service is a core element in OpenStack and is used to authenticate and authorise. It also lists all current services and endpoints. To access Nova for example it must be defined as a service within Keystone. The endpoint provides a Uniform Resource Locator (URL) that provides access to the specific service.

In Keystone, Users and Roles are created and they are assigned to Projects. A Project is typically a customer of OpenStack.

If OpenStack is used as a public cloud, different companies that are purchasing cloud space are distinguished from each-other by the Project which contains the resources used by that customer.

Within a Project environment User accounts are assigned specific Roles and, depending on the role that is assigned to a specific user account, users will have more or less options to do in OpenStack.

Keystone uses a database to store information. The MariaDB database is the default however other databases can be used like Oracle DB or simply an Lightweight Directory Access Protocol (LDAP) directory.
1.3.5 Glance 'Image store' service

Glance image store service is used to store VM disk images. VMs, which are the actual instances are not installed each time, instead they are spawned off from an image. Glance provides the image store. If an administrator wants to boot an instance, then the instance will be booted from the Glance image store.

For scalability, while it is not strictly necessary, the Glance image store typically uses either Swift or Ceph Object Storage as a back-end. In small deployments, it is possible to use local storage as a back-end to Glance, but then everything is bound to the physical server that contains the physical images which is not very scalable.

1.3.6 Other Services

- **Horizon** - Provides the Dashboard, which is a web-based user interface to manage the OpenStack Service
- **Heat** - Provides a service to orchestrate composite cloud applications, using a declarative template format through an OpenStack-native REST API
- **Ceilometer** - It is part of the Telemetry project and provides data collection services for billing and other purposes
- **Trove** - Create and manage databases
- **Sahara** - provides a simple means to provision a data-intensive application cluster
- **Magnum** - provides for Container (CT) orchestration engines such as Docker Swarm, Kubernetes and Apache Mesos. Magnum uses Heat to orchestrate an OS image which contains Docker and Kubernetes and runs that image in either VMs or bare metal in a cluster configuration
- **Ironic** - a bare metal provisioning program and was developed to deploy physical machines (and not VMs).

1.4 Behind the Core OpenStack Projects

To operate effectively the core OpenStack services require some basic services that are vital to their operation.

- **Time synchronisation**
  - This ensures consistent time stamps for communication. Keystone issues tickets that are based on time stamps and without consistent time there will be no communication.
- **Database**
  - By default OpenStack uses the MariaDB database which is used to store all of the cloud-related information.
- **Message queue**
  - The message queue is an essential component that services access to pass messages in an orderly way between services.
1.4.1 The RESTful API

REST is a generic method used to provide access to all the OpenStack components. All of the OpenStack APIs are RESTful, which provides a uniform access. This makes the work of developers much easier, as the same standards are being used throughout.

RESTful API access can be used while implementing commands, but it can also be used directly with **cURL**. For example a *Hypertext Transfer Protocol (HTTP)* POST method and OpenStack will return raw information. Other more friendly methods are available like the command line or the Horizon web interface.

1.5 OpenStack Releases

OpenStack releases are typically twice a year. Each element of a release has its own version number. The current release is **Pike**.

<table>
<thead>
<tr>
<th>Series</th>
<th>Status</th>
<th>Release</th>
<th>EOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queens</td>
<td>Under Development</td>
<td>scheduled</td>
<td>TBD</td>
</tr>
<tr>
<td>Pike</td>
<td>Phase I – Latest release</td>
<td>2017-08-30</td>
<td>TBD</td>
</tr>
<tr>
<td>Ocata</td>
<td>Phase II – Maintained release</td>
<td>2017-02-22</td>
<td>2018-02-26</td>
</tr>
<tr>
<td>Newton</td>
<td>Phase II Maintained Release</td>
<td>2017-08-28</td>
<td>2017-10-11</td>
</tr>
<tr>
<td>Mitaka</td>
<td>Phase III Legacy Release</td>
<td>2016-04-07</td>
<td>2017-04-10</td>
</tr>
<tr>
<td>Liberty</td>
<td>EOL</td>
<td>2015-10-15</td>
<td>2016-11-17</td>
</tr>
<tr>
<td>Kilo</td>
<td>EOL</td>
<td>2015-04-30</td>
<td>2016-05-02</td>
</tr>
<tr>
<td>Juno</td>
<td>EOL</td>
<td>2014-10-16</td>
<td>2015-12-07</td>
</tr>
<tr>
<td>Icehouse</td>
<td>EOL</td>
<td>2014-04-17</td>
<td>2015-07-02</td>
</tr>
<tr>
<td>Havana</td>
<td>EOL</td>
<td>2013-10-17</td>
<td>2014-09-30</td>
</tr>
<tr>
<td>Grizzly</td>
<td>EOL</td>
<td>2013-04-04</td>
<td>2014-03-29</td>
</tr>
<tr>
<td>Folsom</td>
<td>EOL</td>
<td>2012-09-27</td>
<td>2013-11-19</td>
</tr>
<tr>
<td>Essex</td>
<td>EOL</td>
<td>2012-04-05</td>
<td>2013-05-06</td>
</tr>
<tr>
<td>Diablo</td>
<td>EOL</td>
<td>2011-09-22</td>
<td>2013-05-06</td>
</tr>
<tr>
<td>Cactus</td>
<td>Deprecated</td>
<td>2011-04-15</td>
<td></td>
</tr>
<tr>
<td>Bexar</td>
<td>Deprecated</td>
<td>2011-02-03</td>
<td></td>
</tr>
<tr>
<td>Austin</td>
<td>Deprecated</td>
<td>2010-10-21</td>
<td></td>
</tr>
</tbody>
</table>
This page is intentionally blank
2. **OpenStack Training Laboratory**

OpenStack Training Labs is a collection of scripts that install a working OpenStack cluster on a computer. It's an automated, fast, reliable and reproducible way of following OpenStack install guides to create a cluster using KVM/QEMU or VirtualBox VMs.

OpenStack training-labs should run on most common hardware (Desktops/Laptops) out of the box that have a minimum of 8 GB Random Access Memory (RAM). These notes however are working on an assumption of 16 GB RAM with a 1 TB drive.

2.1 **Architecture**

*Illustration 4: OpenStack Laboratory architecture*

Illustration 4 demonstrates the architecture of the OpenStack training laboratory. There are two nodes each with three networks. A Public network attaching the nodes to the Internet, a Management network for internal communications between entities and finally a Provider network to provide an interface for VMs. On the host system it is necessary to implement NAT Masquerade rules to allow the Provider network access the Internet. This can be found in Appendix 1.
2.2 Controller node

The controller node runs the following OpenStack services:

- Keystone, the Identity service.
- Glance, the Image service.
- The management portions of:
  - Nova, the Compute service
  - Neutron, the Networking service
  - Horizon, the Dashboard service.
- OpenStack supporting services like:
  - Structured Query Language (SQL) database
  - Rabbit Message Queue (RabbitMQ)
  - Network Time Protocol (NTP).

The controller node also run optional services like:

- Cinder, Block Storage service
- Swift, Object Storage service
- Heat, Orchestration service
- Telemetry services.

2.3 Compute node

The compute node runs the hypervisor portion of compute that operates instances, this is typically a KVM/QEMU hypervisor. Many compute nodes can be ran to improve scale.

2.3.1 Networking

The compute node also runs a Neutron networking service agent that connects instances to virtual networks and provides firewalling services to instances via security groups.

OpenStack Training Lab scripts automatically create two networks, a Network Management network and a Provider, or External network. These are named differently depending on the hypervisor used.

<table>
<thead>
<tr>
<th>Hypervisor</th>
<th>Management network</th>
<th>Provider network</th>
</tr>
</thead>
<tbody>
<tr>
<td>KVM/QEMU</td>
<td>virbr1</td>
<td>virbr2</td>
</tr>
<tr>
<td>VirtualBox</td>
<td>vboxnet0</td>
<td>vboxnet1</td>
</tr>
</tbody>
</table>
2.4 Passwords

There are many passwords used in this testbed. Here is a simple list of them for reference.

<table>
<thead>
<tr>
<th>Host User</th>
<th>Username</th>
<th>password</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ada Lovelace</td>
<td>alovace</td>
<td>babbage</td>
</tr>
</tbody>
</table>

Note: This represents the user on the host system.

<table>
<thead>
<tr>
<th>Function</th>
<th>Name</th>
<th>Database Pass</th>
<th>Domain Pass</th>
<th>User Pass</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL</td>
<td>root</td>
<td>secrete</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RabbitMQ</td>
<td></td>
<td>rabbitPass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ceilometer</td>
<td>ceilometer</td>
<td>ceilometer_db_secret</td>
<td>ceilometer_user_secret</td>
<td></td>
</tr>
<tr>
<td>Cinder</td>
<td>cinder</td>
<td>cinder_db_secret</td>
<td>cinder_user_secret</td>
<td></td>
</tr>
<tr>
<td>Glance</td>
<td>glance</td>
<td>glance_db_secret</td>
<td>glance_user_secret</td>
<td></td>
</tr>
<tr>
<td>Heat</td>
<td>heat</td>
<td>heat_db_secret</td>
<td>heat_dom_pw</td>
<td>heat_user_secret</td>
</tr>
<tr>
<td>Keystone</td>
<td>keystone</td>
<td>keystone_db_secret</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Neutron</td>
<td>neutron</td>
<td>neutron_db_secret</td>
<td>neutron_user_secret</td>
<td></td>
</tr>
<tr>
<td>Nova</td>
<td>nova</td>
<td>nova_db_secret</td>
<td>nova_user_secret</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Project name</th>
<th>Username</th>
<th>Password</th>
<th>User role name</th>
</tr>
</thead>
<tbody>
<tr>
<td>admin</td>
<td>admin</td>
<td>admin_user_secret</td>
<td>admin</td>
</tr>
<tr>
<td>demo</td>
<td>demo</td>
<td>demo_user_pass</td>
<td>User</td>
</tr>
<tr>
<td>CirrOS VM test</td>
<td>cirros</td>
<td>cubswin:)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Other item</th>
<th>Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Member role name</td>
<td><em>member</em></td>
<td>Member role for generic use</td>
</tr>
<tr>
<td>Service Project name</td>
<td>service</td>
<td></td>
</tr>
<tr>
<td>Mail domain</td>
<td>example.com</td>
<td></td>
</tr>
<tr>
<td>Metadata secret</td>
<td>osbash_training</td>
<td>Used by neutron and nova</td>
</tr>
<tr>
<td>Telemetry secret</td>
<td>osbash_training</td>
<td>Used by ceilometer</td>
</tr>
</tbody>
</table>
This page is intentionally blank
3. **OpenStack training labs pre-installation**

The OpenStack training labs are being continuously updated. It is therefore important to get the latest build. This can be achieved using GIT to clone the latest build.

### 3.1 Get git

Git is a fast, scalable, distributed revision control system. OpenStack training labs builds.

```
ada:~$ sudo apt-get install git
```

### 3.2 Clone the training labs

```
ada:~$ git clone git://git.openstack.org/openstack/training-labs
--branch master
Cloning into 'training-labs'...
remote: Counting objects: 5255, done.
remote: Compressing objects: 100% (1736/1736), done.
remote: Total 5255 (delta 4030), reused 4623 (delta 3468)
Receiving objects: 100% (5255/5255), 988.21 KiB | 217.00 KiB/s, done.
Resolving deltas: 100% (4030/4030), done.
```

Rename to *OpenStack-lab* to be more descriptive.

```
ada:~$ mv training-labs OpenStack-lab
```

### 3.3 Upgrade the training labs

If it is necessary to upgrade the training labs prior to a rebuild of the cluster the enter the ~/OpenStack-lab directory and initiate a git pull.

```
ada:~$ cd OpenStack-lab
ada:~$ cd OpenStack-lab $ git pull
Already up-to-date.
```

### 3.4 Cluster training directory variables

Create a number of variable pointers to the lab root directory ~/OpenStack-lab , ~/OpenStack-lab/labs and ~/OpenStack-lab/labs/osbash for interactive shells. These provide constant pointers no matter where on the filesystem it is chosen to locate the lab.

```
ada:~$ cat <<'EOM' >> ~/.bashrc

OS_LAB=/home/alovelace/OpenStack-lab
OS_ST=/home/alovelace/OpenStack-lab/labs
OS_BASH=/home/alovelace/OpenStack-lab/labs/osbash
EOM
```

02 Oct 2017

OpenStack Pike
Test the variable by running the `~/.bashrc` script again or logging out and back in.

```bash
ada:~$ . ~/.bashrc
ada:~$ echo $OS_LAB
/home/alovelace/OpenStack-lab
ada:~$ echo $OS_ST
/home/alovelace/OpenStack-lab/labs
ada:~$ echo $OS_BASH
/home/alovelace/OpenStack-lab/labs/osbash
```

### 3.5 Pre-installation check

#### 3.5.1 Enable virtualisation support in BIOS

To support Hardware-assisted Virtual Machine (HVM) guests, virtualisation extensions need to be enabled in the Basic Input/Output System (BIOS). In the BIOS the Virtualise option appears under "Advanced Chipset Features" as one of the following:

- Enable Virtualisation Technology - x86 architectures (VT-x).
- Enable Intel VT.
- Vanderpool Technology.

Also enable:

- Intel Virtualisation Technology for Directed I/O (VT-d)

Confirm that the hardware virtualisation is now supported by the Central Processing Unit (CPU) by searching for Virtual Machine eXtensions (VMX) to see if the computer has an Intel processor or Secure Virtual Machine (SVM) for AMD support if the hardware has an AMD processor.

Check that the CPU supports hardware virtualisation. 0 means that the CPU doesn't support hardware virtualisation while > 0 means it does but it still needs to be enabled in the BIOS. (ie. `vmx` or `svm` has appeared x number of times in the output of the command.

```bash
ada:~$ egrep -c '(vmx|svm)' /proc/cpuinfo
4
```

Check if a 64 bit kernel is running. 0 means that the CPU is not 64-bit. Long Mode (LM) equates to a 64-bit CPU.

```bash
ada:~$ egrep -c 'lm' /proc/cpuinfo
8
ada:~$ uname -m
x86_64
```
3.6 Optimise the Nodes

Depending upon available resources on the hypervisor host it is advisable to adjust the memory on the controller node and both the memory and size of the second drive (dev/sdb) in the compute node. The following table outlines the default values for each variable in the $OS_ST/config.

<table>
<thead>
<tr>
<th>Node configuration file</th>
<th>VM Memory</th>
<th>VM CPUs</th>
<th>Second drive</th>
</tr>
</thead>
<tbody>
<tr>
<td>config.controller</td>
<td>5,120 MB</td>
<td>1</td>
<td>N/A</td>
</tr>
<tr>
<td>config.compute1</td>
<td>1,024 MB</td>
<td>1</td>
<td>1,040 MB</td>
</tr>
</tbody>
</table>

In the case of the controller node it runs many services and therefore the demand for memory is high so it is recommend using as much as available on the system. Edit the config.controller file in the $OS_ST/config directory as outlined above. For the compute node, the install guide recommends a minimum is 2048 MB and the default is only 1,024 MB, enough to support 1 instance. The second drive which is distributed between VMs for root disks also needs to be larger. Edit the config.compute1 file in the in the $OS_ST/config directory.

So in an 8 GB system (8,192 MB) the table below are suggested values to adjust in the configuration files. That leaves 1,536 MB for the host system memory.

<table>
<thead>
<tr>
<th>Node configuration file</th>
<th>VM Memory</th>
<th>VM CPUs</th>
<th>Second drive</th>
</tr>
</thead>
<tbody>
<tr>
<td>config.controller</td>
<td>5,120 MB</td>
<td>2</td>
<td>N/A</td>
</tr>
<tr>
<td>config.compute1</td>
<td>1,536 MB</td>
<td>1</td>
<td>25,600 MB</td>
</tr>
</tbody>
</table>

In a 16 GB system (16,384 MB) the table below are suggested values to adjust in the configuration files. That leaves 2,048 MB for the host system memory.

<table>
<thead>
<tr>
<th>Node configuration file</th>
<th>VM Memory</th>
<th>VM CPUs</th>
<th>Second drive</th>
</tr>
</thead>
<tbody>
<tr>
<td>config.controller</td>
<td>6,144 MB</td>
<td>2</td>
<td>N/A</td>
</tr>
<tr>
<td>config.compute1</td>
<td>8,192 MB</td>
<td>1</td>
<td>204,800 MB</td>
</tr>
</tbody>
</table>
3.7 Enable Heat service
By default the Openstack training-labs have the heat service disabled. If the heat service is required carry out this change before running the st.py script to have the service installed.

    ada:~$ cd $OS_ST/config
    ada:~$ sed -i.bak '/heat_controller/s/#//' scripts.ubuntu_cluster

    ada$ diff scripts.ubuntu_cluster.bak scripts.ubuntu_cluster
    45,46c45,46
    < #cmd queue ubuntu/setup_heat_controller.sh
    < #cmd snapshot_cycle -n controller heat_controller_installed
    ---
    > cmd queue ubuntu/setup_heat_controller.sh
    > cmd snapshot_cycle -n controller heat_controller_installed

3.8 Log files
Testbed Log files will be written to the $OS_ST/log directory while the cluster is building.

    ada:~$ ls $OS_ST/log
    000_00_init_controller_node.auto
    001_01_etc_hosts.auto
    002_02_enable_osbash_ssh_keys.auto
    003_03_copy_openrc.auto
    004_04_apt_install_mysql.auto
    005_05_install_rabbitmq.auto
    006_06_install_memcached.auto
    007_07_setup_keystone.auto
    008_08_get_auth_token.auto
    009_09_setup_glance.auto
    010_10_setup_nova_controller.auto
    011_11_setup_neutron_controller.auto
    012_12_setup_self-service_controller.auto
    013_13_setup_neutron_controller_part_2.auto
    014_14_setup_horizon.auto
    015_15_setup_cinder_controller.auto
    016_00_init_compute1_node.auto
    017_01_etc_hosts.auto
    018_02_enable_osbash_ssh_keys.auto
    019_03_copy_openrc.auto
    020_04_setup_nova_compute.auto
    021_05_setup_neutron_compute.auto
    022_06_setup_self-service_compute.auto
    023_07_setup_neutron_compute_part_2.auto
    024_08_setup_cinder_volumes.auto
    025_00_config_public_network.auto
    026_01_config_private_network.auto
    ssh.log
    stacktrain.log
    status
    vboxmanage.log
    vm_compute1.cfg
    vm_controller.cfg
3.9 Add controller and compute1 IP to hypervisor hosts file

Add the Controller and Compute1 IP addresses to the hypervisor `/etc/hosts` file.

```
ada:~$ cat << EOF | sudo tee --append /etc/hosts

# ------------------
# Virtualised nodes
# ------------------

# controller
10.0.0.11   controller

# compute1
10.0.0.31   compute1

EOF

# ------------------
# Virtualised nodes
# ------------------

# controller
10.0.0.11   controller

# compute1
10.0.0.31   compute1
```
This page is intentionally blank
4. Setup OpenStack training labs on KVM/QEMU

The Kernel-based Virtual Machine (KVM) hypervisor is included in major GNU/Linux releases as it has become the hypervisor of choice within the GNU/Linux community therefore it is available within the various distribution repositories. In actual fact KVM is a GNU/Linux kernel module that permits programs within user space to access either the Intel or AMD processor virtualisation features. As a result KVM VMs actually run as user space processes. KVM uses QEMU, a generic and open source machine emulator and virtualiser for Input/Output (I/O) hardware emulation. It can emulate a variety of processors on a guest processor and combined with the KVM kernel module it can approach native speeds. All combinations of 32-bit and 64-bit host and guest systems are supported, except 64-bit guests on 32-bit hosts.

KVM is managed via the libvirt API and tools such as virsh, virtinstall, virt-clone, virt-viewer and virt-manager.

KVM is a Type-1 hypervisor that runs directly on x86 hardware. The GNU/Linux interface makes it look like it is a hosted hypervisor running on it, but infact each VM is running on the bare metal with the host GNU/Linux OS providing a launchpad for the hypervisor and then engaging in a co-processing relationship with the hypervisor.

On x86 hardware, KVM relies on the hardware virtualisation instructions that are embedded in the processors and therefore these advanced chipset features must be enabled. Using these instructions the hypervisor and each guest VM run directly on the bare metal, and most of the resource translations are performed by the hardware.

Libvirt provides command line tools under the virsh root command while virt-manager provides a graphical tool. This lab in the main operates in headless mode, i.e. it doesn't require the graphical tool.
4.1 Installation
KVM requires a number of elements to operate.

4.1.1 Install KVM packages

- **libvirt** - a C toolkit to interact with the virtualisation capabilities of GNU/Linux. The library provides a C API for different virtualisation mechanisms and currently supports QEMU, KVM, XEN, OpenVZ, LXC, and VirtualBox.
- **qemu-kvm** - permits the running of multiple virtual computers, each running unmodified GNU/Linux or Windows images on X86 hardware. Each VM has private virtualised hardware: a network card, disk, graphics adapter, etc.
- **libguestfs-tools** - library that allows access and modification to guest disk images.
- **virt-manager** - graphical user interface.

```
ada:~$ sudo apt-get install qemu-kvm libvirt-bin libguestfs-tools virt-manager
[sudo] password for alovleace: babbage
```

4.2 GNU/Linux Bridge utilities

Without a bridge KVM VMs will only have network access to other VMs on the same server and to the host itself via a shared private network 192.168.122.0. To allow VMs access to the LAN, create a network bridge on the host.

```
ada:~$ sudo apt-get install bridge-utils
ada:~$ sudo usermod -aG libvirt `id -un`
```

4.3 virt-manager

It may appear strange but it is important to run the **virt-manager**. This triggers QEMU to create a default pool for storage. As the server is headless this must be performed using Secure SHell (SSH) X11 forwarding.

SSH to the host using the following switches.

- **-M** Places the SSH client into *master* mode for connection sharing.
- **-Y** Enables trusted X11 forwarding. Trusted X11 forwardings are not subjected to the X11 SECURITY extension controls.
Set the `virsh` default connect URI which eliminates the need to use the long-winded `virsh` connection command to the KVM/QEMU hypervisor. Enable by running the file.

```
ada:~$ cat << EOM >> ~/.bashrc

# Variable to set virsh default URI to QEMU
VIRSH_DEFAULT_CONNECT_URI='qemu:///system'
EOM

ada:~$ . ~/.bashrc

ada:~$ echo $VIRSH_DEFAULT_CONNECT_URI
qemu:///system
```

Now connect to the `virsh` shell on the KVM/QMEU hypervisor.

```
ada:~$ virsh

virsh # uri
qemu:///system
```

It is also possible to run virsh commands directly from the shell.

```
ada:~$ virsh uri
qemu:///system
```
Check that a default storage pool exists.

```
virsh # pool-list
Name     State  Autostart
default  active yes
```

4.4 Build introduction

The cluster is built in three phases:

- Download the OS image.
- Build a base disk, about 40 to 50 minutes.
- Build the controller and compute1 node VMs based on the base disk, about 50 to 65 minutes.

Essentially the scripts download the Ubuntu image, run it up on a KVM/QEMU VM with some configuration files. The script recovers the Internet Protocol (IP) address of the base and connects to it over SSH on the standard port 22. It upgrades the VM and installs the relevant OpenStack cluster software. After the base disk, the command builds two node VMs (controller and compute) from it.

If you have a previous build and find that it is necessary to rebuild the base disk, simply delete the disk file in the $OS_ST/img directory. That will force the download of a new base disk otherwise if the script finds an existing base disk it will simply bypass that step and go about building the controller and compute nodes and initial configuration as summarised in the build steps.
4.5 Build steps

Controller node

- Edit the `/etc/hosts` file
- Enable osbash SSH keys
- Install mysql
- Install rabbitmq
- Install memcached
- Setup keystone
- Setup Glance
- Setup Nova controller
- Setup Neutron controller
- Setup self-service controller
- Setup Horizon
- Setup Cinder controller
- Setup Heat controller (If enabled in `scripts.ubuntu_cluster` file).

Compute node

- Setup Nova compute
- Setup Neutron compute
- Setup self-service compute
- Setup Cinder volumes.

Controller node

- Configure public network
- Configure private network.
4.6 Run the stacktrain script

4.6.1 Stacktrain

The *stacktrain* python script installs the training-labs.

```
ada:~$ cd $OS_ST
ada:~/OpenStack-lab/labs $./st.py --build cluster --provider kvm
INFO Using provider kvm.
INFO stacktrain start at Sat Sep 23 22:31:11 2017
INFO Asked to delete VM base.
INFO not found
WARNING There is no file at given path:
/home/alovelace/OpenStack-lab/labs/img/ubuntu-16.04.3-server-amd64.iso
INFO Downloading
   http://releases.ubuntu.com/16.04/ubuntu-16.04.3-server-amd64.iso
to /home/alovelace/OpenStack-lab/labs/img/ubuntu-16.04.3-server-
amd64.iso
INFO This may take a while.
INFO Download succeeded.
....

See Appendix 6 - stacktrain cluster creation script – KVM for detail.
....

INFO Processing of scripts successful.
INFO Cluster build took 2006 seconds
```

4.6.2 Confirm installed release

Confirm the installed release version.

```
ada:~$ grep OPENSTACK_RELEASE $OS_ST/config/openstack
: ${OPENSTACK_RELEASE:=pike}
```

4.6.3 Memory and harddisks

Confirm that the memory and hard-disks are the sizes configured in the
*config.controller* and *config.compute1* files.

**Controller**

```
osbash@controller:~$ cat /proc/meminfo | grep MemTotal
MemTotal: 6110832 kB

osbash@controller:~$ df -h | grep ^/dev
/dev/sda1 9.0G 2.7G 5.9G 31% /

osbash@controller:~$ lsblk
NAME MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda  8:0  0  9.8G  0 disk
|-sda1  8:1  0  9.3G  0 part /
|-sda2  8:2  0  1K  0 part
`-sda5  8:5  0  510M  0 part [SWAP]
```
Compute1

```bash
osbash@compute1:~$ cat /proc/meminfo | grep MemTotal
MemTotal:                   16432844 kB
```

```bash
osbash@compute1:~$ df -h | grep ^/dev
/dev/sda1       9.0G  2.4G  6.2G  29% /
```

```bash
osbash@compute1:~$ lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda      8:0    0 9.8G 0 disk
 |-sda1   8:1    0 9.3G 0 part /
 |-sda2   8:2    0  1K 0 part
 `-sda5   8:5    0 510M 0 part [SWAP]
sdb     8:16   0 200G 0 disk
```

4.7 Using the cluster

By default, the cluster is built in headless mode. As such, the method to access the cluster nodes is via SSH. Get the IP addresses for each node on the virbr0 bridge interface to access for management.

Access the shell prompts on the cluster nodes using SSH where the username is `osbash` and the password is also `osbash`. To become root, use `sudo`.

Optionally it is possible to use `virt-manager` to access via the GUI tool.

4.7.1 Review the running VMs

```bash
ada:~$ virsh
virsh # list
   Id    Name                           State
-----------------------------------------------
    3    compute1                       running
   25    controller                     running
```

4.7.2 Controller node

```bash
virsh # dominfo controller
   Id:    7
   Name:  controller
   UUID:  8b3ecf79-b414-453e-927a-2887377bdcee
   OS Type:  hvm
   State:  running
   CPU(s):  2
   CPU time:  695.8s
   Max memory:  6291456 KiB
   Used memory:  6291456 KiB
   Persistent:  yes
   Autostart:  disable
   Managed save:  no
   Security model:  apparmor
   Security DOI:  0
   Security label:  libvirt-8b3ecf79-b414-453e-927a-2887377bdcee (enforcing)
```

```bash
virsh # snapshot-list controller
   Name                 Creation Time             State
-----------------------------------------------
controller_-_cluster_installed 2017-09-24 00:35:09 +0300 shutoff
```

02 Oct 2017 OpenStack Pike netLabs!UG
4.7.3 Compute node

```
virsh # dominfo compute1
Id:             8
Name:           compute1
UUID:           7344fcf1-0155-4da1-a53b-6bc7daf86d59
OS Type:        hvm
State:          running
CPU(s):         1
CPU time:       344.5s
Max memory:     16777216 KiB
Used memory:    16777216 KiB
Persistent:     yes
Autostart:      disable
Managed save:   no
Security model: apparmor
Security DOI:   0
Security label: libvirt-7344fcf1-0155-4da1-a53b-6bc7daf86d59
=enforcing
```

```
virsh # snapshot-list compute1
Name                 Creation Time             State
------------------------------------------------------------
```

4.7.4 VM IP addresses

The VM IP addresses on the public network are given at the end of the `stacktrain` script.

Your cluster nodes:

- **INFO** VM name: `compute1`
  - SSH login: `ssh osbash@192.168.122.71`
    - (password: `osbash`)
- **INFO** VM name: `controller`
  - SSH login: `ssh osbash@192.168.122.205`
    - (password: `osbash`)
  - Dashboard: Assuming horizon is on controller VM.
  - User : `demo` (password: `demo_user_pass`)
  - User : `admin` (password: `admin_user_secret`)
- **INFO** Network: mgmt
- **INFO** Network address: 10.0.0.0
- **INFO** Network: provider
- **INFO** Network address: 203.0.113.0

It is also possible from the hypervisor to access the VMs over the management network.

- **VM name:** `compute1`
  - SSH login: `ssh osbash@10.0.0.11` (password: `osbash`)

- **VM name:** `controller`
  - SSH login: `ssh osbash@10.0.0.31` (password: `osbash`)

---

**netLabs!UG**  
OpenStack Pike  
02 Oct 2017
### 4.8 Reviewing the networks created by the script

During the execution of the scripts two new networks are created. *labs-mgmt* (virbr1) is a *management network* using IP addresses from the private IP address space 10.0.0.0/24 and *labs-provider* (virbr2) is the *provider network* using addresses from the 203.0.113.0/24 subnet. It is from this range of IP addresses that VM instances will receive floating IP addresses.

```
virsh # net-list
Name                  State  Autostart Persistent
----------------------------------------------
default               active  yes       yes
labs-mgmt             active  no        yes
labs-provider         active  no        yes
```

```
virsh # net-dumpxml labs-mgmt
<network connections='2'>
  <name>labs-mgmt</name>
  <uuid>70cb6b14-8e78-4a71-9589-725dbd7ef018</uuid>
  <forward mode='nat'>
    <nat>
      <port start='1024' end='65535'/>
    </nat>
  </forward>
  <bridge name='virbr1' stp='on' delay='0'/>
  <mac address='52:54:00:b9:7c:fc'/>
  <ip address='10.0.0.1' netmask='255.255.255.0'/>
</network>
```

```
virsh # net-dumpxml labs-provider
<network connections='2'>
  <name>labs-provider</name>
  <uuid>5f8f547b-246b-4698-9c6b-1c8db221e26d</uuid>
  <forward mode='nat'>
    <nat>
      <port start='1024' end='65535'/>
    </nat>
  </forward>
  <bridge name='virbr2' stp='on' delay='0'/>
  <mac address='52:54:00:52:f6:de'/>
  <ip address='203.0.113.1' netmask='255.255.255.0'/>
</network>
```
4.9 Access the Controller node

ada:~$ ssh osbash@10.0.0.11
The authenticity of host '10.0.0.11 (10.0.0.11)' can't be established.
ECDSA key fingerprint is
SHA256:flu02u/jXomo3bmh/WE4h2Abqka5bmb1Xv1fQP2mc.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.0.31' (ECDSA) to the list of known hosts.
osbash@10.0.0.31's password: osbash
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-96-generic x86_64)

* Documentation:  https://help.ubuntu.com
* Management:     https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage

4.10 Access the Compute node

ada:~$ ssh osbash@10.0.0.31
The authenticity of host '10.0.0.31 (10.0.0.31)' can't be established.
ECDSA key fingerprint is
SHA256:flu02u/jXomo3bmh/WE4h2Abqka5bmb1Xv1fQP2mc.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.0.31' (ECDSA) to the list of known hosts.
osbash@10.0.0.31's password: osbash
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-96-generic x86_64)

* Documentation:  https://help.ubuntu.com
* Management:     https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage
4.11 Add hypervisor SSH keys to the controller and compute1 nodes

Optionally add SSH host keys from the hypervisor to the Controller and Compute1 nodes. This removes the need for passwords when logging in to the nodes from the hypervisor.

```
ada:~$ ssh-keygen -t rsa -b 4096 -C "ada@lovelace.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/alovelace/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/alovelace/.ssh/id_rsa.
Your public key has been saved in /home/alovelace/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:Y24YPdnqY3TK36Bi2KESL6DdKGrjd7oUqf10LOZr4pA ada@lovelace.com
The key's randomart image is:
+---[RSA 4096]----+
|                 |
| o . o           |
| . = . o=+.      |
| |E B B.*+o.     |
||ooBoXo*o=.. o   |
|>|=o+**=.ooo. .  |
+----[SHA256]-----+
```

```
ada:~$ ssh-agent
SSH_AUTH_SOCK=/tmp/ssh-GW8hKy5WuK2Z/agent.7155; export SSH_AUTH_SOCK;
SSH_AGENT_PID=7156; export SSH_AGENT_PID;
```

```
ada:~$ ssh-copy-id osbash@controller
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed:
"/home/alovelace/.ssh/id_rsa.pub"
```

```
ada:~$ ssh-copy-id osbash@compute1
```

```
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed:
"/home/alovelace/.ssh/id_rsa.pub"
```

```
The authenticity of host 'compute1 (10.0.0.31)' can't be established.
ECDSA key fingerprint is
SHA256:xRkeJHD8dTRggBZ+NdNAb7WuJ3qJQqm5B81zvqH4uvE.
Are you sure you want to continue connecting (yes/no)? yes
```

```
Number of key(s) added: 1
Now try logging into the machine, with:   "ssh 'osbash@controller''
and check to make sure that only the key(s) you wanted were added.
```

```
Number of key(s) added: 1
Now try logging into the machine, with:   "ssh 'osbash@compute1''
and check to make sure that only the key(s) you wanted were added.
```
This page is intentionally blank
5. **Setup OpenStack training labs on VirtualBox**

VirtualBox is a cross-platform virtualisation application, a type-2 hypervisor that allows for the running of multiple operating systems simultaneously. The only practical limits are disk space and memory. VirtualBox can run everywhere from small embedded systems or desktop class machines all the way up to data centre deployments and even Cloud environments. In this OpenStack Lab one can run multiple VM instances simultaneously. This allows for the testing of OpenStack and the lab servers and their harddisks can be arbitrarily frozen, woken up, copied, backed up, and transported between hosts.

VirtualBox provides both a graphical tool for managing VMs as well as a fully featured set of shell commands under a root command `vboxmanage`. For the most part this lab will use the VMs in headless mode, in other words without the need for the graphical tool.

```
ada:~$ sudo apt-get install virtualbox
[sudo] password for alovelace: babbage
```
5.1 Build introduction
The cluster is built in three phases:

- Download the OS image.
- Build a base disk, about 30 to 40 minutes.
- Build the controller and compute1 node VMs based on the base disk, about 25 to 30 minutes.

By default, the cluster is built on VirtualBox hypervisor.

Essentially the scripts download the Ubuntu image, run it up on VirtualBox with some configuration files such that it can login to the base VM using SSH on port 2229. It then upgrades the VM and and installs the relevant OpenStack cluster software. After the base disk, the command builds two node VMs (controller and compute) from it.

If you have a previous build and find that it is necessary to rebuild the base disk, simply delete the disk file in the $OS_ST/img directory. That will force the download of a new base disk otherwise if the script finds an existing base disk it will simply bypass that step and go about building the controller and compute1 nodes and initial configuration as summarised in the build steps.

5.2 Build steps

Controller node

- Install mysql
- Install rabbitmq
- Install memcached
- Setup keystone
- Setup Glance
- Setup Nova controller
- Setup Neutron controller
- Setup self-service controller
- Setup Horizon
- Setup Cinder controller
- Setup Heat controller (If enabled in scripts.ubuntu_cluster file).

Compute node

- Setup Nova compute
- Setup Neutron compute
- Setup self-service compute
- Setup Cinder volumes.

Controller node

- Configure public network
- Configure private network.
5.3 Run the scripts

5.3.1 Stacktrain

The `stacktrain` python script installs the training-labs.

    ada:~/OpenStack-lab/labs $ cd $OS_ST/
    ada:~$ ./st.py --build cluster

INFO Using provider virtualbox.
INFO stacktrain start at Fri Sep 22 16:24:57 2017
INFO Creating
    /home/alovelace/OpenStack-lab/labs/img/base-ssh-pike-ubuntu-16.04-amd64.vdi.
INFO ISO image okay.
INFO Install ISO:
    /home/alovelace/OpenStack-lab/labs/img/ubuntu-16.04.3-server-amd64.iso
INFO Asked to delete VM base
INFO not found
INFO Created VM base.
....

See Appendix 7 - stacktrain cluster creation script - VirtualBox for detail.
....

INFO Processing of scripts successful.
INFO Cluster build took 1037 seconds

5.3.2 Confirm installed release

Confirm the installed release version.

    ada:~$ grep OPENSTACK_RELEASE $OS_ST/config/openstack:
    : ${OPENSTACK_RELEASE=:pike}

5.3.3 Memory and harddisks

Confirm that the memory and hard-disks are the sizes configured in the `config.controller` and `config.compute1` files.

**Controller**

    osbash@controller:~$ cat /proc/meminfo | grep MemTotal
    MemTotal:  6110832 kB

    osbash@controller:~$ df -h | grep ^/dev
    /dev/sda1 9.0G 2.7G 5.9G 31% /

    osbash@controller:~$ lsblk
    NAME   MAJ:MIN   RM  SIZE RO TYPE MOUNTPOINT
    sda     8:0       0  9.8G  0 disk
    |-'sda1  8:1       0  9.3G  0 part /
    |-'sda2  8:2       0  1K   0 part
    `-'sda5  8:5       0 510M  0 part [SWAP]
Compute1

osbash@compute1:~$ cat /proc/meminfo | grep MemTotal
MemTotal: 8175396 kB

osbash@compute1:~$ df -h | grep ^/dev
/dev/sda1  9.0G  2.4G  6.2G  28% /

osbash@compute1:~$ lsblk
NAME   MAJ:MIN  RM  SIZE RO TYPE MOUNTPOINT
sda      8:0    0  9.8G  0 disk
|---sda1  8:1    0  9.3G  0 part /
|---sda2  8:2    0  1K   0 part
`---sda5  8:5    0  510M  0 part [SWAP]
sdb      8:16   0  50G   0 disk

5.4 Using the cluster

By default, the cluster is built in headless mode. As such, the method to access the cluster nodes is via SSH. The localhost's Transmission Control Protocol (TCP) ports 2230 through 2232 are forwarded to the nodes.

Access the shell prompts on the cluster nodes as follows. The username is osbash and the password is also osbash. To become root, use sudo.

Optionally it is possible to use either the VirtualBox Graphical tool or indeed install the VirtualBox phpVirtualBox web-based front-end to Oracle VirtualBox to manage and administer VMs via a web browser. This is particularly useful if the host is a VM with a cloud provider.

5.4.1 Review the running VMs

ada:~$ vboxmanage list runningvms
"controller" {a698b5ae-1bc0-4cbe-897e-8e741970be7a}
"compute1" {ff1c0b3e-fe09-435c-9b17-089d28fd3bf6}
5.4.2 Controller node

```bash
ada:\$ vboxmanage showvminfo "controller"
Name: controller
Groups: /labs
Guest OS: Ubuntu (64-bit)
UUID: a698b5ae-1bc0-4cbe-897e-8e741970be7a
Config file: /home/dobriain/VirtualBox
VMs/labs/controller/controller.vbox
Snapshot folder: /home/dobriain/VirtualBox
VMs/labs/controller/Snapshots
Log folder: /home/dobriain/VirtualBox VMs/labs/controller/Logs
Hardware UUID: a698b5ae-1bc0-4cbe-897e-8e741970be7a
Memory size: 6144MB
Page Fusion: off
VRAM size: 8MB
CPU exec cap: 100%
HPET: off
Chipset: piix3
Firmware: BIOS
Number of CPUs: 2
PAE: on
Long Mode: on
CPUID Portability Level: 0
CPUID overrides: None
Boot menu mode: disabled
Boot Device (1): HardDisk
Boot Device (2): DVD
Boot Device (3): Network
Boot Device (4): Not Assigned
ACPI: on
IOAPIC: on
Time offset: 0ms
RTC: UTC
Hardw. virt.ext: on
Nested Paging: on
Large Pages: on
VT-x VPID: on
VT-x unr. exec.: on
Paravirt. Provider: Default
State: running (since 2017-09-25T13:08:04.167000000)
Monitor count: 1
3D Acceleration: off
2D Video Acceleration: off
Teleporter Enabled: off
Teleporter Port: 0
Teleporter Address: 
Teleporter Password: 
Tracing Enabled: off
Allow Tracing to Access VM: off
Tracing Configuration:
Autostart Enabled: off
Autostart Delay: 0
Default Frontend:
Storage Controller Name (0): SATA
Storage Controller Type (0): IntelAhci
Storage Controller Instance Number (0): 0
Storage Controller Max Port Count (0): 30
Storage Controller Port Count (0): 3
Storage Controller Bootable (0): on
Storage Controller Name (1): IDE
Storage Controller Type (1): PIIX4
```
OpenStack Training Laboratory

Storage Controller Instance Number (1): 0
Storage Controller Max Port Count (1): 2
Storage Controller Port Count (1): 2
Storage Controller Bootable (1): on
SATA (0, 0): /home/dobriain/VirtualBox VMs/labs/controller/Snapshots/{6ef3f63b-0e33-4bbf-9d71-c8f071f7a037}.vdi (UUID: 6ef3f63b-0e33-4bbf-9d71-c8f071f7a037)
NIC 1:  MAC: 0800276D5F96, Attachment: NAT, Cable connected: on, Trace: off (file: none), Type: virtio, Reported speed: 0 Mbps, Boot priority: 0, Promisc Policy: deny, Bandwidth group: none
NIC 1 Settings: MTU: 0, Socket (send: 64, receive: 64), TCP Window (send:64, receive: 64)
NIC 1 Rule(0): name = http, protocol = tcp, host ip = 127.0.0.1, host port = 8888, guest ip = , guest port = 80
NIC 1 Rule(1): name = ssh, protocol = tcp, host ip = 127.0.0.1, host port = 2230, guest ip = , guest port = 22
NIC 2:  MAC: 080027008BD4, Attachment: Host-only Interface 'vboxnet4', Cable connected: on, Trace: off (file: none), Type: virtio, Reported speed: 0 Mbps, Boot priority: 1, Promisc Policy: allow-all, Bandwidth group: none
NIC 3:  MAC: 08002798C487, Attachment: Host-only Interface 'vboxnet5', Cable connected: on, Trace: off (file: none), Type: virtio, Reported speed: 0 Mbps, Boot priority: 0, Promisc Policy: allow-all, Bandwidth group: none
NIC 4:  disabled
NIC 5:  disabled
NIC 6:  disabled
NIC 7:  disabled
NIC 8:  disabled
Pointing Device: PS/2 Mouse
Keyboard Device: PS/2 Keyboard
UART 1:  disabled
UART 2:  disabled
UART 3:  disabled
UART 4:  disabled
LPT 1:  disabled
LPT 2:  disabled
Audio:  disabled
Clipboard Mode: disabled
Drag and drop Mode: disabled
Session name: headless
Video mode: 640x480x8 at 0,0 enabled
VRDE:  disabled
USB:  disabled
EHCI:  disabled
XHCI:  disabled
USB Device Filters:
<none>
Available remote USB devices:
<none>
Currently Attached USB Devices:
<none>
Bandwidth groups: <none>
Shared folders: <none>
VRDE Connection: not active
Clients so far: 0

Video capturing: not active
Capture screens: 0
Capture file: /home/dobriain/VirtualBox
VMs/labs/controller/controller.webm
Capture dimensions: 1024x768
Capture rate: 512 kbps
Capture FPS: 25

Guest:

Configured memory balloon size: 0 MB
OS type: Linux26_64
Additions run level: 1
Additions version: 5.0.18_Ubuntu r106667

Guest Facilities:

Facility "VirtualBox Base Driver": active/running (last update: 2017/09/25 13:08:13 UTC)
Facility "Seamless Mode": not active (last update: 2017/09/25 13:08:04 UTC)

Snapshots:

Name: controller_-_cluster_installed (UUID: 3d80bede-33c2-4787-a38d-38a62ba3f672) *
5.4.3 Compute node

```
ada:~$ vboxmanage showvminfo "compute1"
Name:            compute1
Groups:          /labs
Guest OS:        Ubuntu (64-bit)
UUID:            ff1c0b3e-fe09-435c-9b17-089d28fd3bf6
Config file:     /home/dobriain/VirtualBox
VMs/labs/compute1/compute1.vbox
Snapshot folder: /home/dobriain/VirtualBox VMs/labs/compute1/Snapshots
Log folder:      /home/dobriain/VirtualBox VMs/labs/compute1/Logs
Hardware UUID:   ff1c0b3e-fe09-435c-9b17-089d28fd3bf6
Memory size:     8192MB
Page Fusion:     off
VRAM size:       8MB
CPU exec cap:    100%
HPET:            off
Chipset:         piix3
Firmware:        BIOS
Number of CPUs:  1
PAE:             on
Long Mode:       on
CPUID Portability Level: 0
CPUID overrides: None
Boot menu mode:  disabled
Boot Device (1): HardDisk
Boot Device (2): DVD
Boot Device (3): Network
Boot Device (4): Not Assigned
ACPI:            on
IOAPIC:          on
Time offset:     0ms
RTC:             UTC
Hardw. virt.ext: on
Nested Paging:   on
Large Pages:     on
VT-x VPID:       on
VT-x unr. exec.: on
Paravirt. Provider: Default
State:           running (since 2017-09-25T13:08:04.753000000)
Monitor count:   1
3D Acceleration: off
2D Video Acceleration: off
Teleporter Enabled: off
Teleporter Port:  0
Teleporter Address: 
Teleporter Password: 
Tracing Enabled: off
Allow Tracing to Access VM: off
Tracing Configuration: 
Autostart Enabled: off
Autostart Delay:  0
Default Frontend: 
Storage Controller Name (0): SATA
Storage Controller Type (0): IntelAhci
Storage Controller Instance Number (0): 0
Storage Controller Max Port Count (0): 30
Storage Controller Port Count (0):  3
Storage Controller Bootable (0): on
Storage Controller Name (1): IDE
Storage Controller Type (1): PIIX4
Storage Controller Instance Number (1): 0
Storage Controller Max Port Count (1): 2
```
<table>
<thead>
<tr>
<th>Storage Controller Port Count (1):</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>SATA (0, 0): /home/dobriain/VirtualBox VMs/labs/compute1/Snapshots/ (ea022ebc-d776-4827-aaf3-b105672f2df2).vdi (UUID: ea022ebc-d776-4827-aaf3-b105672f2df2)</td>
<td></td>
</tr>
<tr>
<td>SATA (1, 0): /home/dobriain/VirtualBox VMs/labs/compute1/Snapshots/ (f1d957c9-6a75-4f42-879e-4749496d5798).vdi (UUID: f1d957c9-6a75-4f42-879e-4749496d5798)</td>
<td></td>
</tr>
<tr>
<td>NIC 1: MAC: 080027E225DB, Attachment: NAT, Cable connected: on, Trace: off (file: none), Type: virtio, Reported speed: 0 Mbps, Boot priority: 0, Promisc Policy: deny, Bandwidth group: none</td>
<td></td>
</tr>
<tr>
<td>NIC 1 Settings: MTU: 0, Socket (send: 64, receive: 64), TCP Window (send:64, receive: 64)</td>
<td></td>
</tr>
<tr>
<td>NIC 1 Rule(0): name = ssh, protocol = tcp, host ip = 127.0.0.1, host port = 2232, guest ip = , guest port = 22</td>
<td></td>
</tr>
<tr>
<td>NIC 2: MAC: 0800271784D5, Attachment: Host-only Interface 'vboxnet4', Cable connected: on, Trace: off (file: none), Type: virtio, Reported speed: 0 Mbps, Boot priority: 1, Promisc Policy: allow-all, Bandwidth group: none</td>
<td></td>
</tr>
<tr>
<td>NIC 3: MAC: 080027874221, Attachment: Host-only Interface 'vboxnet5', Cable connected: on, Trace: off (file: none), Type: virtio, Reported speed: 0 Mbps, Boot priority: 0, Promisc Policy: allow-all, Bandwidth group: none</td>
<td></td>
</tr>
<tr>
<td>NIC 4: disabled</td>
<td></td>
</tr>
<tr>
<td>NIC 5: disabled</td>
<td></td>
</tr>
<tr>
<td>NIC 6: disabled</td>
<td></td>
</tr>
<tr>
<td>NIC 7: disabled</td>
<td></td>
</tr>
<tr>
<td>NIC 8: disabled</td>
<td></td>
</tr>
<tr>
<td>Pointing Device: PS/2 Mouse</td>
<td></td>
</tr>
<tr>
<td>Keyboard Device: PS/2 Keyboard</td>
<td></td>
</tr>
<tr>
<td>UART 1: disabled</td>
<td></td>
</tr>
<tr>
<td>UART 2: disabled</td>
<td></td>
</tr>
<tr>
<td>UART 3: disabled</td>
<td></td>
</tr>
<tr>
<td>UART 4: disabled</td>
<td></td>
</tr>
<tr>
<td>LPT 1: disabled</td>
<td></td>
</tr>
<tr>
<td>LPT 2: disabled</td>
<td></td>
</tr>
<tr>
<td>Audio: disabled</td>
<td></td>
</tr>
<tr>
<td>Clipboard Mode: disabled</td>
<td></td>
</tr>
<tr>
<td>Drag and drop Mode: disabled</td>
<td></td>
</tr>
<tr>
<td>Session name: headless</td>
<td></td>
</tr>
<tr>
<td>Video mode: 640x480x8 at 0,0 enabled</td>
<td></td>
</tr>
<tr>
<td>VRDE: disabled</td>
<td></td>
</tr>
<tr>
<td>USB: disabled</td>
<td></td>
</tr>
<tr>
<td>EHCI: disabled</td>
<td></td>
</tr>
<tr>
<td>XHCI: disabled</td>
<td></td>
</tr>
</tbody>
</table>

USB Device Filters: <none>

Available remote USB devices: <none>

Currently Attached USB Devices: <none>

Bandwidth groups: <none>

Shared folders: <none>

VRDE Connection: not active

Clients so far: 0
Video capturing: not active
Capture screens: 0
Capture file: /home/dobriain/VirtualBox VMs/labs/compute1/compute1.webm
Capture dimensions: 1024x768
Capture rate: 512 kbps
Capture FPS: 25

Guest:
Configured memory balloon size: 0 MB
OS type: Linux26_64
Additions run level: 1
Additions version: 5.0.18_Ubuntu r106667

Guest Facilities:
Facility "VirtualBox Base Driver": active/running (last update: 2017/09/25 13:08:20 UTC)
Facility "Seamless Mode": not active (last update: 2017/09/25 13:08:04 UTC)

Snapshots:
Name: compute-_cluster_installed (UUID: 26f08d75-930d-495d-ba6a-eb0f2e62c0cc) *

5.4.4 VM IP addresses

The VM IP addresses on the public network are given at the end of the stacktrain script.

Your cluster nodes:

Your cluster nodes:
INFO VM name: compute1
INFO SSH login: ssh -p 2232 osbash@127.0.0.1 (or localhost)
INFO (password: osbash)
INFO VM name: controller
INFO SSH login: ssh -p 2230 osbash@127.0.0.1 (or localhost)
INFO (password: osbash)
INFO Dashboard: Assuming horizon is on controller VM.
INFO http://127.0.0.1:8888/horizon/
INFO User: demo (password: demo_user_pass)
INFO User: admin (password: admin_user_secret)
INFO Network: mgmt
INFO Network address: 10.0.0.0
INFO Network: provider
INFO Network address: 203.0.113.0
5.5 Reviewing the networks created by the script

During the execution of the scripts two networks are created. `vboxnet0` is a management network using IP addresses from the private IP address space 10.0.0.0/24 and `vboxnet1` is the provider network using addresses from the 203.0.113.0/24 subnet. It is from this range of IP addresses that VM instances will receive floating IP addresses.

```
ada:~$ vboxmanage list hostonlyifs
Name: vboxnet0
GUID: 786f6276-656e-4074-8000-0a0027000000
DHCP: Disabled
IPAddress: 10.0.0.1
NetworkMask: 255.255.255.0
IPV6Address: fe80:0000:0000:0000:0800:27ff:fe00:0000
IPV6NetworkMaskPrefixLength: 64
HardwareAddress: 0a:00:27:00:00:00
MediumType: Ethernet
Status: Up
VBoxNetworkName: HostInterfaceNetworking-vboxnet0

Name: vboxnet1
GUID: 786f6276-656e-4174-8000-0a0027000001
DHCP: Disabled
IPAddress: 203.0.113.1
NetworkMask: 255.255.255.0
IPV6Address: fe80:0000:0000:0000:0800:27ff:fe00:0001
IPV6NetworkMaskPrefixLength: 64
HardwareAddress: 0a:00:27:00:00:01
MediumType: Ethernet
Status: Up
VBoxNetworkName: HostInterfaceNetworking-vboxnet1
```

5.6 Access the Controller node

```
ada:~$ ssh -p 2230 osbash@localhost
The authenticity of host 'localhost:2230 ([127.0.0.1]:2230)' can't be established.
ECDSA key fingerprint is SHA256:lafd719aAi9CHkvKOsdvhRHvX/KkRkk7i8zg9gjiU.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[localhost]:2230' (ECDSA) to the list of known hosts.
osbash@localhost's password: osbash
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-77-generic x86_64)

* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage
```
5.7 Access the Compute node

`ada:~$ ssh -p 2232 osbash@localhost`

The authenticity of host '[localhost]:2232 ([127.0.0.1]:2232)' can't be established.
ECDSA key fingerprint is
SHA256:lafd719aAi9CHkvK0sdvHhX/KkRkx7i8zq09q1U.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[localhost]:2232' (ECDSA) to the list of known hosts.
osbash@localhost's password: osbash
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-77-generic x86_64)

* Documentation:  https://help.ubuntu.com
* Management:     https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage

5.8 Add hypervisor SSH keys to the controller and compute1 nodes

Optionally add SSH host keys from the hypervisor to the Controller and Compute1 nodes. This removes the need for passwords when logging in to the nodes from the hypervisor.

`ada:~$ ssh-keygen -t rsa -b 4096 -C "ada@lovelace.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/alovelace/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/alovelace/.ssh/id_rsa.
Your public key has been saved in /home/alovelace/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:Y24YPdnqY3TK36Bi2KESl6DdGRyd70Uqf10LOZr4pA ada@lovelace.com
The key's randomart image is:
++--[RSA 4096]-----
|               |
|   o .. .    |
| .. = . o=+=.|
| . E B B *+=0.|
| ooBoX0*0= . 0|
| =0**=.0oo .. |
++--[SHA256]-----

`ada:~$ ssh-agent`
SSH_AUTH_SOCK=/tmp/ssh-GW8hKy5WuK2Z/agent.7155; export SSH_AUTH_SOCK;
SSH_AGENT_PID=7156; export SSH_AGENT_PID;
echo Agent pid 7156;

`ada:~$ ssh-copy-id osbash@controller`
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed:
"/home/alovelace/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
osbash@controller's password: osbash
Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'osbash@controller'" and check to make sure that only the key(s) you wanted were added.
ada:~$ ssh-copy-id osbash@compute1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed:
"/home/alovelace/.ssh/id_rsa.pub"
The authenticity of host 'compute1 (10.0.0.31)' can't be established.
ECDSA key fingerprint is
SHA256:xRxeJHD8dTRggBZ+NdNAb7WuJ3qJQqm5B8izvqH4uvE.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to
filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you
are prompted now it is to install the new keys
osbash@compute1's password: osbash

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'osbash@compute1'"
and check to make sure that only the key(s) you wanted were added.
This page is intentionally blank
6. Operating the OpenStack training testbed on KVM/QEMU

6.1 Managing KVM/QEMU VMs in headless mode

To manage the two VMs in headless mode, on KVM/QEMU it is necessary to learn a few basic virsh commands.

6.2 Starting the VMs

List the available VMs and check if they are running.

```
ada:~$ virsh
Welcome to virsh, the virtualization interactive terminal.
Type: 'help' for help with commands
'quit' to quit

virsh # list --all
Id    Name                           State
----------------------------------------------------
3    compute1                       shut off
2    controller                     shut off
```

Start the VMs.

```
virsh # start controller
WDomain controller started

virsh # start compute1
Domain compute1 started
```

Check the VMs are running.

```
virsh # list
Id    Name                           State
----------------------------------------------------
3    compute1                       running
2    controller                     running
```

6.3 Powering off the VMs

Power off the VMs as follows.

```
virsh # shutdown controller
Domain controller is being shutdown

virsh # list
Id    Name                           State
----------------------------------------------------
3    compute1                       running
```
6.4 Saving VM state and stopping

Save the state of a VM and then stop it by executing the command:

```bash
class Name: Compute1

# save

virsh save compute1 compute1.dump

# list

domain compute1 saved to compute1.dump

# list

virsh list --all

Id   Name                        State
----------------------------------
3    compute1                   shut off
2    controller                 shut off
```

6.5 Managing snapshots

List the snapshots associated with a VM.

```bash
class Name: Controller

# snapshot-list

virsh snapshot-list controller

Name                      Creation Time               State
----------------------------------
controller_-_cluster_installed 2017-09-24 01:24:57 +0300 shutoff
```

6.6 Taking a snapshot

It is best to shutdown the VM first because a snapshot taken of a running guest only captures the state of the disk and not the state of the memory.

```bash
class Name: Controller

# shutdown

virsh shutdown controller

Domain controller is being shutdown

# snapshot-create-as

virsh snapshot-create-as --domain controller
--name "snap01-controller" --disk-only --atomic
--diskspec hda,file=/var/lib/libvirt/images/snap01-controller

Domain snapshot snap01-controller created

# snapshot-list

virsh snapshot-list controller

Name                      Creation Time               State
----------------------------------
1493825797 2017-09-24 18:36:37 +0300 running
controller_-_cluster_installed 2017-09-24 01:24:57 +0300 shutoff
snap01-controller 2017-09-24 18:37:07 +0300 disk-snapshot
```

6.7 Restoring to a previous snapshot

To restore to a previous snapshot, in this case `controller_-_cluster_installed` just:

```bash
class Name: Controller

# snapshot-revert

virsh snapshot-revert --domain controller --snapshotname controller_-_cluster_installed --running

# list

virsh list

Id   Name               State
----------------------------------
3    controller         running
6.8 Delete a snapshot

Delete a snapshot by:

```
virsh # snapshot-delete --domain controller --snapshotname snap01-controller
```

Error: Failed to delete snapshot snap01-controller
Error: unsupported configuration: deletion of 1 external disk snapshots not supported yet

However as can be seen deletion of external disk snapshots is not supported yet. In this case delete the metadata associated with the snapshot and delete the snapshot manually.

```
virsh # snapshot-delete --domain controller --metadata snap01-controller
```

Domain snapshot snap01-controller deleted

```
ada:~$ sudo rm /var/lib/libvirt/images/snap01-controller
```

```
virsh # snapshot-list controller
Name                 Creation Time             State
------------------------------------------------------------
1493825797           2017-09-24 18:36:37 +0300 running
controller_-_cluster_installed 2017-09-24 01:24:57 +0300 shutoff
```

6.9 Increase the size of the Compute1 node

If the nodes were not optimised as described in the setup chapters, it is possible to change their size afterwards. The default compute1 node second drive and RAM are both quite small for all but the smallest images. In its default configuration it has 1 GB of memory, 1 CPU and 1 GB volume for Cinder Block Storage service. (Note: /dev/sda is for the VM itself and /dev/sdb is assigned to LVM for Cinder).

Here is an example of how the CPUs can be increased to 2, memory to 4 GB and /dev/sdb to 20 GB.

First shut down the compute1 VM instance and confirm it has shutdown.

```
virsh # shutdown compute1
Domain compute1 is being shutdown
```

```
virsh # list --all
Id   Name                        State
-------------------------------------------
3    controller                  running
-    compute1                   shut off
```

Edit the VM instance XML file and change the maximum and current memory to 17 GB and CPUs to 2.

```
virsh # edit compute1
...<memory unit='KiB'>17825792</memory>
<currentMemory unit='KiB'>17825792</currentMemory>
<vcpu placement='static'>2</vcpu>
...```
Have a look at the `compute1-sdb` image as it is.

```bash
ada:~$ sudo qemu-img info /var/lib/libvirt/images/compute1-sdb
image: /var/lib/libvirt/images/compute1-sdb
  file format: qcow2
  virtual size: 200G (214748364800 bytes)
  disk size: 1.0G
  cluster_size: 65536
  Format specific information:
    compat: 0.10
    refcount bits: 16
```

Now resize the QEMU QCOW Image by adding 50G to bring the image up to 20G.

```bash
ada:~$ sudo qemu-img resize /var/lib/libvirt/images/compute1-sdb +50G
Image resized.
```

Start the VM instance.

```bash
virsh # start compute1
Domain compute1 started
```

Connect to the `compute1` node and review the reported size of the physical volume `/dev/sdb` by LVM, it is still 1 GB.

```bash
osbash@compute1:~$ sudo pvdisplay
--- Physical volume ---
PV Name           /dev/sdb
VG Name           cinder-volumes
PV Size           200.00 GiB / not usable 4.00 MiB
Allocatable       yes
PE Size           4.00 MiB
Total PE          51199
Free PE           51199
Allocated PE      0
PV UUID           uRKbME-24kE-1iHL-paym-TPgg-lelk-8OpH1D
```

However `fdisk` reports its true size.

```bash
osbash@compute1:~$ sudo fdisk -l /dev/sdb
Disk /dev/sdb: 250 GiB, 268435456000 bytes, 524288000 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

Physical Volume Show (pvs) command reports information about physical volumes, it also considers it still 1 GB.

```bash
osbash@compute1:~$ sudo pvs /dev/sdb
PV  VG  Fmt  Attr  PSize  PFree
/dev/sdb  cinder-volumes  lvm2  a--  200.00g  200.00g
```
Resize the LVM physical volume by forcing LVM to re-evaluate the reported size in the actual image file.

```
osbash@compute1:~$ sudo pvresize /dev/sdb
Physical volume "/dev/sdb" changed
1 physical volume(s) resized / 0 physical volume(s) not resized
```

Confirm the change.

```
osbash@compute1:~$ sudo pvs /dev/sdb
PV VG Fmt Attr PSize PFree
/dev/sdb cinder-volumes lvm2 a-- 250.00g 250.00g
```

```
osbash@compute1:~$ sudo pvdisplay
--- Physical volume ---
PV Name /dev/sdb
VG Name cinder-volumes
PV Size 250.00 GiB / not usable 3.00 MiB
Allocatable yes
PE Size 4.00 MiB
Total PE 63999
Free PE 63999
Allocated PE 0
PV UUID uRKbME-24kE-1iHL-paym-TPgg-1elk-8opH1D
```

```
osbash@compute1:~$ sudo vgdisplay
--- Volume group ---
VG Name cinder-volumes
System ID
Format lvm2
Metadata Areas 1
Metadata Sequence No 4
VG Access read/write
VG Status resizable
MAX LV 0
Cur LV 0
Open LV 0
Max PV 0
Cur PV 1
Act PV 1
VG Size 250.00 GiB
PE Size 4.00 MiB
Total PE 63999
Alloc PE / Size 0 / 0
Free PE / Size 63999 / 250.00 GiB
VG UUID 7YI12b-ewAV-BT8j-pydq-5kyo-z6Yn-2d1fV0
```

Confirm the `nova-compute` service is operational.

```
osbash@controller:~$ . admin-openrc.sh
osbash@controller:~$ openstack compute service list
+----+------------------+------------+----------+---------+-------+----------------------------+
| ID | Binary           | Host       | Zone     | Status  | State | Updated At                 |
+----+------------------+------------+----------+---------+-------+----------------------------+
|  1 | nova-scheduler   | controller | internal | enabled | up    | 2017-09-24T12:11:19.000000 |
|  5 | nova-consoleauth | controller | internal | enabled | up    | 2017-09-24T12:11:15.000000 |
|  6 | nova-conductor   | controller | internal | enabled | up    | 2017-09-24T12:11:22.000000 |
|  8 | nova-compute     | compute1   | nova     | enabled | up    | 2017-09-24T12:11:17.000000 |
+----+------------------+------------+----------+---------+-------+----------------------------+
```
This page is intentionally blank
7. **Operating the OpenStack training testbed on VirtualBox**

7.1 **Managing VirtualBox VMs in headless mode**

To manage the two VMs in headless mode, on VirtualBox it is necessary to learn a few basic `vboxmanage` commands.

7.2 **Starting the VMs**

List the available VMs and check if they are running.

```
ada:~$ vboxmanage list vms
"controller" {a698b5ae-1bc0-4cbe-897e-8e741970be7a}
"compute1" {ff1c0b3e-fe09-435c-9b17-089d28fd3bf6}
```

Start the VMs.

```
ada:~$ vboxmanage startvm "controller" --type headless
Waiting for VM "controller" to power on...
VM "controller" has been successfully started.
```

```
ada:~$ vboxmanage startvm "compute1" --type headless
Waiting for VM "compute1" to power on...
VM "compute1" has been successfully started.
```

Check the VMs are running.

```
ada:~$ vboxmanage list runningvms
"controller" {a698b5ae-1bc0-4cbe-897e-8e741970be7a}
"compute1" {ff1c0b3e-fe09-435c-9b17-089d28fd3bf6}
```

7.3 **Powering off the VMs**

Power off the VMs as follows.

```
ada:~$ vboxmanage controlvm "controller" poweroff
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
```

```
ada:~$ vboxmanage list runningvms
"compute1" {ff1c0b3e-fe09-435c-9b17-089d28fd3bf6}
```

7.4 **Saving VM state and stopping**

Save the state of a VM and then stop it execute command:

```
ada:~$ vboxmanage controlvm "compute1" savestate
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
```

```
ada:~$ vboxmanage list runningvms
```

7.5 Managing snapshots
List the snapshots associated with a VM.

```bash
ada:~$ vboxmanage snapshot "controller" list
  Name: controller_-_cluster_installed (UUID: b445f1e1-9d87-4eeb-8a12-63396456d190) *
```

7.6 Taking a snapshot

```bash
ada:~$ vboxmanage snapshot "controller" take "snap01-controller"
   --description "Initial controller snapshot"
   0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
   Snapshot taken. UUID: 798b7138-6802-49dc-b1c3-86ed7406417f

ada:~$ vboxmanage snapshot "controller" list
  Name: controller_-_cluster_installed
       (UUID: b445f1e1-9d87-4eeb-8a12-63396456d190)
  Name: snap01-controller
       (UUID: 798b7138-6802-49dc-b1c3-86ed7406417f) *
  Description: Initial Controller snapshot
```

7.7 Restoring to a previous snapshot
To restore to a previous snapshot, in this case `controller_-_cluster_installed` just power down the VM, restore the VM and restart.

```bash
ada:~$ vboxmanage controlvm "controller" poweroff
   0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%

ada:~$ vboxmanage snapshot "controller" restore controller_-_cluster_installed
   Restoring snapshot b445f1e1-9d87-4eeb-8a12-63396456d190
   0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%

ada:~$ vboxmanage startvm "controller" -type headless
Waiting for VM "controller" to power on...
VM "controller" has been successfully started.

ada:~$ vboxmanage startvm "controller" -type headless
Waiting for VM "controller" to power on...
VM "controller" has been successfully started.

Notice that the asterisk (*) has moved to the active snapshot.

```bash
ada:~$ vboxmanage snapshot "controller" list
  Name: controller_-_cluster_installed
       (UUID: b445f1e1-9d87-4eeb-8a12-63396456d190) *
  Name: snap01-controller
       (UUID: 798b7138-6802-49dc-b1c3-86ed7406417f)
  Description: Initial Controller snapshot
```

Delete a snapshot
Delete a snapshot and notice it removed from the snapshot list.

```bash
ada:~$ vboxmanage snapshot "controller" delete snap01-controller
   0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%

ada:~$ vboxmanage snapshot "controller" list
  Name: controller_-_cluster_installed (UUID: b445f1e1-9d87-4eeb-8a12-63396456d190) *
```
7.8 Increase the size of the Compute1 node

If the nodes were not optimised as described in the setup chapters, it is possible to change their size afterwards. The default compute1 node second drive and RAM are both quite small for all but the smallest images. In its default configuration it has 1 GB of memory, 1 CPU and 1 GB volume for Cinder Block Storage service. (Note: /dev/sda is for the VM itself and /dev/sdb is assigned to LVM for Cinder).

Consider firstly the current state of the compute1 node.

```
1
```

```
osbash@compute1:~$ cat /proc/cpuinfo | grep processor | wc -l
1
```

```
osbash@controller:~$ cat /proc/meminfo | grep MemTotal
MemTotal: 8175264 kB
```

```
osbash@controller:~$ lsblk
NAME   MAJ:MIN   RM SIZE RO TYPE MOUNTPOINT
sda      8:0   0   9.8G  0 disk
 |-sda1   8:1   0   9.3G  0 part /
 |-sda2   8:2   0   1K  0 part
`-sda5   8:5   0   510M  0 part [SWAP]
sdb 8:16   0   50G  0 disk
```

```
ada:$ ls ~/VirtualBox VMs/labs
compute1 controller
```

```
ada:$ ls ~/VirtualBox VMs/labs/compute1
compute1.vbox compute1.vbox-prev Logs Snapshots
```

There is 1 CPU, 8 GB memory and a 50 GB second drive (/dev/sdb). As an example increase the CPUs can be increased to 2, memory to 9 GB and /dev/sdb to 60 GB.

First shut down the compute1 VM instance and confirm it has shutdown.

```
ada:$ vboxmanage controlvm "compute1" poweroff
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
```

```
Edit the VM instance memory to 9 GB and CPUs to 2.
```

```
ada:$ vboxmanage modifyvm "compute1" --cpus 2
ada:$ vboxmanage modifyvm "compute1" --memory 9216
```

As the working image is made up of the base image and the snapshots it is necessary to clone the VM instance to create a new base.

```
ada:$ vboxmanage clonemvm "compute1" 0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
Machine has been successfully cloned as "compute1 Clone"
```

```
ada:$ ls ~/VirtualBox VMs/compute1 Clone/
compute1 Clone-disk1.vdi compute1 Clone-disk2.vdi compute1 Clone.vbox
```
Delete the original `compute1` instance VM.

```
ada:~$ vboxmanage unregistervm "compute1" --delete
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
```

```
ada:~$ ls ~/VirtualBox VMs/labs
controller
```

Move the new clone directory to the place previously taken by `compute1`.

```
ada:~$ cd ~/'VirtualBox VMs'
ada:~/VirtualBox VMs $ mv 'compute1 Clone' labs/
ada:~/VirtualBox VMs $ cd labs
ada:~/VirtualBox VMs/labs $ mv 'compute1 Clone' compute1
ada:~/VirtualBox VMs $ cd compute1
```

Rename the clone files.

```
ada:~/VirtualBox VMs/labs/compute1 $ mv 'compute1 Clone-disk1.vdi' compute1-disk1.vdi
ada:~/VirtualBox VMs/labs/compute1 $ mv 'compute1 Clone-disk2.vdi' compute1-disk2.vdi
ada:~/VirtualBox VMs/labs/compute1 $ mv 'compute1 Clone.vbox' compute1.vbox
```

```
ada:~/VirtualBox VMs/labs/compute1 $ ls
compute1-disk1.vdi  compute1-disk2.vdi  compute1.vbox
```

```
Edit the vbox file to reflect the new `compute1` name and update the vdi names.
```
```
ada:~/VirtualBox VMs/labs/compute1 $ sed -i.bak 's/compute1 Clone/compute1/' compute1.vbox
ada:~/VirtualBox VMs/labs/compute1 $ diff compute1.vbox.bak compute1.vbox
```

```
9c9
<   <Machine uuid="{42d461ef-79cf-49a7-a6fd-5bcfcafdcd87c}" name="compute1 Clone" OSType="Ubuntu_64" snapshotFolder="Snapshots" currentStateModified="false" lastStateChange="2017-09-24T20:51:23Z">
---
>   <Machine uuid="{42d461ef-79cf-49a7-a6fd-5bcfcafdcd87c}" name="compute1" OSType="Ubuntu_64" snapshotFolder="Snapshots" currentStateModified="false" lastStateChange="2017-09-24T20:51:23Z">
```

```
12,13c12,13
<         <HardDisk uuid="{5259ea5f-d2ca-402b-99ba-48cb4199b451}" location="compute1 Clone-disk1.vdi" format="VDI" type="Normal"/>
<         <HardDisk uuid="{5ead5e53-593b-4eba-a228-7d513751beec}" location="compute1 Clone-disk2.vdi" format="VDI" type="Normal"/>
---
>         <HardDisk uuid="{5259ea5f-d2ca-402b-99ba-48cb4199b451}" location="compute1-disk1.vdi" format="VDI" type="Normal"/>
>         <HardDisk uuid="{5ead5e53-593b-4eba-a228-7d513751beec}" location="compute1-disk2.vdi" format="VDI" type="Normal"/>
```
Register the `compute1` VM instance.

```
ada:~$ vboxmanage registervm /home/alovelace/’VirtualBox VMs’/labs/compute1/compute1.vbox
```

Confirm registration.

```
ada:~$ vboxmanage list vms
"controller" {85cc5cd8-3392-49bd-bac8-76c4a8bed317}
"compute1" {42d461ef-79cf-49a7-a6fd-5bcfcafc87c}
```

Have a look at the `compute1-disk2` image as it is, note the size is 1040 MB.

```
ada:~$ vboxmanage list hdds | awk -v RS='' '/base/'
UUID:           6a0cfecf-fd21-42b0-b91f-58bd7f44c871
Parent UUID:    base
State:          locked read
Type:           multiattach
Location:       /home/alovelace/Dropbox/OpenStack-lab/labs/img/base-ssh-ocata-ubuntu-16.04-amd64.vdi
Storage format: VDI
Capacity:       10000 MBytes
Encryption:     disabled
```

Now resize the image to 60G.

```
ada:~$ vboxmanage modifymedium 5ead5e53-593b-4eba-a228-7d513751beec
--resize 61440
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
```

Confirm the change.

```
ada:~$ vboxmanage list hdds | awk -v RS='' '/base/'
UUID: 6a0cfecf-fd21-42b0-b91f-58bd7f44c871
Parent UUID: base
State: locked read
Type: multiattach
Location: /home/alovelace/OpenStack-lab/labs/img/base-ssh-ocata-ubuntu-16.04-amd64.vdi
Storage format: VDI
Capacity: 10000 MBytes
Encryption: disabled
```

```
ada:~$ vboxmanage list hdds | awk -v RS='' '/base/'
UUID: 5259ea5f-d2ca-402b-99ba-48cb4199b451
Parent UUID: base
State: created
Type: normal (base)
Location: /home/alovelace/VirtualBox VMs/labs/compute1/compute1-disk1.vdi
Storage format: VDI
Capacity: 10000 MBytes
Encryption: disabled
```

```
ada:~$ vboxmanage list hdds | awk -v RS='' '/base/'
UUID: 5ead5e53-593b-4eba-a228-7d513751beec
Parent UUID: base
State: created
Type: normal (base)
Location: /home/alovelace/VirtualBox VMs/labs/compute1/compute1-disk2.vdi
Storage format: VDI
Capacity: 61440 MBytes
Encryption: disabled
```

Start the VM instance.

```
ada:~$ vboxmanage startvm "compute1" --type headless
Waiting for VM "compute1" to power on...
VM "compute1" has been successfully started.
```

Connect to the compute1 node and review.

```
osbash@compute1:~$ cat /proc/cpuinfo | grep processor | wc -l
2
```

```
osbash@controller:~$ cat /proc/meminfo | grep MemTotal
MemTotal: 9207452 kB
```

```
osbash@controller:~$ lsblk
NAME  MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda   8:0  0  9.8G  0 disk
   |-sda1 8:1  0  9.3G  0 part /
   |-sda2 8:2  0  1K  0 part
   `--sda5 8:5  0  510M  0 part [SWAP]
sdb   8:16  0  60G  0 disk
```
However the reported size of the physical volume /dev/sdb by LVM, it is still 50 GB.

```bash
osbash@compute1:~$ sudo pvdisplay
--- Physical volume ---
PV Name       /dev/sdb
VG Name       cinder-volumes
PV Size       50.00 GiB / not usable 4.00 MiB
Allocatable   yes
PE Size       4.00 MiB
Total PE      12799
Free PE       12799
Allocated PE  0
PV UUID       9XFbcy-WuIl-hBoa-4L4i-SvyL-ay30-M9zfLc
```

However like `lsblk`, `fdisk` reports its true size.

```bash
osbash@compute1:~$ sudo fdisk -l /dev/sdb
Disk /dev/sdb: 60 GiB, 64424509440 bytes, 125829120 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

Physical Volume Show (pvs) command reports information about physical volumes, it also considers it still 1 GB.

```bash
osbash@compute1:~$ sudo pvs /dev/sdb
PV       VG             Fmt  Attr PSize  PFree
/dev/sdb  cinder-volumes lvm2 a-- 50.00g 50.00g
```

Resize the LVM physical volume by forcing LVM to re-evaluate the reported size in the actual image file.

```bash
osbash@compute1:~$ sudo pvresize /dev/sdb
Physical volume "/dev/sdb" changed
1 physical volume(s) resized / 0 physical volume(s) not resized
```

Confirm the change.

```bash
osbash@compute1:~$ sudo pvs /dev/sdb
PV       VG             Fmt  Attr PSize  PFree
/dev/sdb  cinder-volumes lvm2 a-- 60.00g 60.00g
```

```bash
osbash@compute1:~$ sudo pvdisplay
--- Physical volume ---
PV Name       /dev/sdb
VG Name       cinder-volumes
PV Size       60.00 GiB / not usable 3.00 MiB
Allocatable   yes
PE Size       4.00 MiB
Total PE      15359
Free PE       15359
Allocated PE  0
PV UUID       9XFbcy-WuIl-hBoa-4L4i-SvyL-ay30-M9zfLc
```
osbash@compute1:~$ sudo vgdisplay
--- Volume group ---
VG Name               cinder-volumes
System ID
Format                lvm2
Metadata Areas        1
Metadata Sequence No  4
VG Access             read/write
VG Status             resizable
MAX LV                0
Cur LV                0
Open LV               0
Max PV                0
Cur PV                1
Act PV                1
VG Size               60.00 GiB
PE Size               4.00 MiB
Total PE              15359
Alloc PE / Size       0 / 0
Free  PE / Size       15359 / 60.00 GiB
VG UUID               ql3SLC-S7Vq-9zVK-dDw9-722w-Ycv2-b28Xp7

Confirm the nova-compute service is operational on the controller node.

osbash@controller:~$ . admin-openrc.sh
osbash@controller:~$ openstack compute service list
+----+------------------+------------+----------+---------+-------+----------------------------+
| ID | Binary           | Host       | Zone     | Status  | State | Updated At                 |
+----+------------------+------------+----------+---------+-------+----------------------------+
|  1 | nova-scheduler   | controller | internal | enabled | up    | 2017-09-24T21:22:51.000000 |
|  2 | nova-consoleauth | controller | internal | enabled | up    | 2017-09-24T21:22:56.000000 |
|  6 | nova-conductor   | controller | internal | enabled | up    | 2017-09-24T21:22:47.000000 |
|  7 | nova-compute     | compute1   | nova     | enabled | up    | 2017-09-24T21:22:54.000000 |
8. **Reviewing the Installation**

8.1 **Controller node - Database**

Access the MariaDB database as the database *root* user. Note that since Ubuntu 16.04 access to the database requires *sudo* privileges. This is because plugin value for the *root* user is set to *unix_socket* and this socket is only accessible by the Operating System root user. All other users have a blank plugin value which defaults to *mysql_native_password*.

Access the controller node shell.

```bash
osbash@controller:~$ sudo mysql -u root -p
Enter password: secret
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 228
Server version: 10.0.28-MariaDB-0ubuntu0.16.04.1 Ubuntu 16.04

Copyright (c) 2000, 2016, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

Here is the changed setting since Ubuntu 16.04.

```sql
MariaDB [(none)]> SELECT user,plugin FROM mysql.user;
+----------+-------------+
| user     | plugin      |
+----------+-------------+
| root     | unix_socket |
| keystone |             |
| keystone |             |
| glance   |             |
| glance   |             |
| nova     |             |
| nova     |             |
| neutron  |             |
| neutron  |             |
| cinder   |             |
| cinder   |             |
| heat     |             |
| heat     |             |
+----------+-------------+
13 rows in set (0.00 sec)

MariaDB [(none)]>
```
It is also possible to see a list of databases within the MariaDB.

```
MariaDB [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| cinder             |
| glance             |
| heat               |
| information_schema |
| keystone           |
| mysql              |
| neutron            |
| nova               |
| nova_api           |
| nova_cell0         |
| performance_schema |
+--------------------+
10 rows in set (0.04 sec)

MariaDB [(none)]> exit
Bye
```

Now using the username and database password for one of the services, say **keystone** review the database tables. Note: **sudo** is not necessary.

```
osbash@controller:~$ mysql -u keystone -p
Enter password: keystone_db_secret
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 115
Server version: 10.0.29-MariaDB-0ubuntu0.16.04.1 Ubuntu 16.04
Copyright (c) 2000, 2016, Oracle, MariaDB Corporation Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

Now listing the databases, only the **keystone** one is available to this user.

```
MariaDB [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| keystone           |
+--------------------+
2 rows in set (0.00 sec)
```
Change to that database and review the tables within.

MariaDB [(none)]> USE keystone;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed

MariaDB [keystone]> SHOW TABLES;
+------------------------+
| Tables_in_keystone     |
+------------------------+
| access_token           |
| assignment             |
| config_register        |
| consumer               |
| credential             |
| endpoint               |
| endpoint_group         |
| federated_user         |
| federation_protocol    |
| group                  |
| id_mapping             |
| identity_provider      |
| idp_remote_ids         |
| implied_role           |
| local_user             |
| mapping                |
| migrate_version        |
| nonlocal_user          |
| password               |
| policy                 |
| policy_association     |
| project                |
| project_endpoint       |
| project_endpoint_group |
| region                 |
| request_token          |
| revocation_event       |
| role                   |
| sensitive_config       |
| service                |
| service_provider       |
| token                  |
| trust                  |
| trust_role             |
| user                   |
| user_group_membership  |
| user_option            |
| whitelisted_config     |
+------------------------+
38 rows in set (0.00 sec)
### 8.2 Client environment scripts

To increase efficiency of client operations, OpenStack supports simple client environment scripts also known as OpenRC files. These scripts typically contain common options for all clients, but also support unique options. They are on the root of each VM called:

- `admin-openrc.sh`
- `demo-openrc.sh`

Review the scripts.

```bash
osbash@controller:~$ cat admin-openrc.sh
export OS_USERNAME=admin
export OS_PASSWORD=admin_user_secret
export OS_PROJECT_NAME=admin
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_DOMAIN_NAME=Default
export OS_AUTH_URL=http://10.0.0.11:35357/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2

guest@controller:~$ cat demo-openrc.sh
export OS_USERNAME=demo
export OS_PASSWORD=demo_user_pass
export OS_PROJECT_NAME=demo
export OS_USER_DOMAIN_NAME=default
export OS_PROJECT_DOMAIN_NAME=default
export OS_AUTH_URL=http://10.0.0.11:5000/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
```
8.3 Identity Service - Keystone

The Identity Service provides the following functions:

- **User management**: Tracks users and their permissions.
- **Service catalogue**: A catalogue of available services and their API endpoints.
- **User**: Digital representation of a person, system or service who uses OpenStack cloud services. The Identity Service validates that incoming requests are made by the *User* who claims to be making the call. Users have a login and may be assigned a token to access resources. Users can be directly assigned to a particular project and behave as if they are contained within that project (note in earlier versions projects were called tenants).
- **Credentials**: Data that is known only to a particular user for the purpose of identifying themselves, proving who they are. Examples are Username/Password, Username/API key or authentication token.
- **Authentication**: The act of confirming the identity of a *User* by validating the credentials of that user. These are initially a Username/Password or Username/API key and the Identity Service issues an authentication token to the User which can be provided by the User for subsequent requests.
- **Token**: An arbitrary piece of text used to access resources. Each token has a scope, describing what resources are accessible with it. Token can be revoked at any time and are valid for a finite duration. More protocols will be supported in the future.
- **Domain**: An Identity API v3 entity. Represents a collection of *Projects*, *Groups* and *Users* that define administrative boundaries for managing OpenStack Identity entities.
- **Project**: A container used to group or isolate resources and/or identity objects. Depending on the service operator, a project may map to a customer, account, organisation or project.
- **Service**: A service such as Compute (Nova), Object storage (Swift) or Image service (Glance). It provides one or more endpoints through which users can access resources and perform operations.
- **Endpoint**: A network accessible address, usually described by a URL, from where you a service can be accessed.
- **Role**: A personality that a *User* assumes that enables them to perform a specific set of operations. A *Role* includes a set of rights and privileges. A User assuming that Role inherits those rights and privileges. In the Identity Service, a *Token* that is issued to a User includes a list of Roles that a User has. Services that are being called by that User determine how they interpret the set of Roles a User has and to which operations or resources each Role grants access.

8.3.1 Controller node

Review the *domain*, *project*, *user*, and *roles*. Make sure to run the *admin* OpenRC script first to set the admin variables.

```
osbash@controller:~$ . admin-openrc.sh
```
bash@controller:~$ openstack domain list
+--------------------------------+---------+---------+--------------------------+
| ID                             | Name    | Enabled | Description              |
+--------------------------------+---------+---------+--------------------------+
| 34db125c6a77495695aea728c5ad5f32 | heat    | True    | Stack projects and users |
| default                        | Default | True    | The default domain       |
+--------------------------------+---------+---------+--------------------------+

bash@controller:~$ openstack project list
+--------------------------------+---------+
| ID                             | Name    |
+--------------------------------+---------+
| 24789cdee2c0499c86735b8558b6fa0f | demo    |
| 666e2166b56a4d56a9df82c30b53b076 | admin   |
| 828a05665714667875dd3c3719bfc00 | service |
+--------------------------------+---------+

bash@controller:~$ openstack user list
+-----------------------------------+-------------------+
| ID                               | Name              |
+-----------------------------------+-------------------+
| 3fc3608c8fdd472bf507d03fa5dc42d2 | demo              |
| 42a30bf1acb64c44bc1ca5fcd034b20 | nova              |
| 48e6666c81d0466391ae912e29ad579 | heat              |
| 5651003d94ad44300a7033ec9bb74656 | cinder            |
| 5c2873a66a562854778f66fb69966 | heat_domain_admin |
| 5e066502514481e865e0ed17b7d7941 | neutron           |
| 9d46e35f8ada4696b23b9a816ddcb8 | glance            |
| ced53a38098d4904a9acfa4736c527eb | admin             |
| ddf90b302f0e44d984e82bc9935220e5 | placement         |
+-----------------------------------+-------------------+

bash@controller:~$ openstack role list
+-----------------------------------+------------------+
| ID                               | Name             |
+-----------------------------------+------------------+
| 38b157ee745746a7b7598e926dd72cc8 | admin            |
| 9fe2f99ea4384b894a90878d3e92bab | _member_         |
| ba13400433342b5a0cfeedb3bfe4a9d | heat_stack_owner |
| bf3915a6f6154f07aed78f0c2497b0b1 | user             |
| f37b6133a764247bc84db4d3e7fe85 | heat_stack_user  |
+-----------------------------------+------------------+
8.4 Image Service - Glance

The OpenStack Image service enables users to discover, register and retrieve VM images. Glance offers an RESTful API that enables querying VM metadata and retrieval of the image. It supports the storage of disk or server images on various repository types, including OpenStack Object Storage.

The OpenStack Image service includes the following components:

- **glance-api**
  - Accepts Image API calls for image discovery, retrieval, and storage.

- **glance-registry**
  - Stores, processes, and retrieves metadata about images. Metadata includes items such as size and type.

- **Database**
  - Stores image metadata and you can choose your database depending on your preference. Most deployments use MySQL or SQLite.

- **Storage repository**
  - Various repository types are supported including normal file systems, Object Storage, Reliable Autonomic Distributed Object Store (RADOS) block devices, HTTP, and Amazon S3. Note that some repositories will only support read-only usage.

8.4.1 Controller node

Review existing images and add a new image.

```
$ openstack image list
```

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>d40c0820-b8f2-4e33-9f58-6f590709c01a</td>
<td>cirros</td>
<td>active</td>
</tr>
<tr>
<td>d40c0820-b8f2-4e33-9f58-6f590709c01a</td>
<td>cirros</td>
<td>active</td>
</tr>
</tbody>
</table>

02 Oct 2017 OpenStack Pike
8.5 Compute service - Nova

OpenStack Compute hosts and manages cloud computing systems. It interacts with OpenStack Identity for authentication, OpenStack Image service for disk and server images, and OpenStack Dashboard for the User and administrative interface. Image access is limited by Projects, and by Users; quotas are limited per project (the number of instances, for example). OpenStack Compute can scale horizontally on standard hardware, and download images to launch instances.

OpenStack Compute consists of the following areas and their components:

8.5.1 API

- **nova-api service**
  - Accepts and responds to end user compute API calls. The service supports the OpenStack Compute API, the Amazon Elastic Compute 2 (EC2) API, and a special Admin API for privileged Users to perform administrative actions. It enforces some policies and initiates most orchestration activities, such as running an instance.

- **nova-api-metadata service**
  - Accepts metadata requests from instances. The nova-api-metadata service is generally used when you run in multi-host mode with nova-network installations.

8.5.2 Compute core

- **nova-compute service**
  - A worker daemon that creates and terminates VM instances through hypervisor APIs. For example:
    - XenAPI for XenServer/Xen Cloud Platform (XCP)
    - libvirt for KVM or QEMU
    - VMwareAPI for VMware.

- **nova-scheduler service**
  - Takes a VM instance request from the queue and determines on which compute server host it runs.

- **nova-conductor module**
  - Mediates interactions between the nova-compute service and the database. It eliminates direct accesses to the cloud database made by the nova-compute service. The nova-conductor module scales horizontally. However, do not deploy it on nodes where the nova-compute service runs.

8.5.3 Networking for VMs

- **nova-network worker daemon**
  - Similar to the nova-compute service, accepts networking tasks from the queue and manipulates the network. Performs tasks such as setting up bridging interfaces or changing IPtables rules.
8.5.4 Console interface

- **nova-novncproxy daemon**
  - Provides a proxy for accessing running instances through a Virtual Network Computing (VNC) connection. Supports browser-based novnc clients.

- **nova-spicehtml5proxy daemon**
  - Provides a proxy for accessing running instances through a Simple Protocol for Independent Computing Environments (SPICE) connection. Supports browser-based HTML5 client.

- **nova-xvpvncproxy daemon**
  - Provides a proxy for accessing running instances through a VNC connection. Supports an OpenStack specific Java client.

- **nova-cert module**
  - A server daemon that serves the Nova Cert service for X509 certificates. It is used to generate certificates for euca-bundle-image. Only needed for the EC2 API.

- **nova-cert daemon**
  - x509 certificates.

8.5.5 Image management

- **euca2ools client**
  - A set of command-line interpreter commands for managing cloud resources. Although it is not an OpenStack module, you can configure nova-api to support this EC2 interface.

8.5.6 Command-line clients and other interfaces

- **nova client**
  - Enables users to submit commands as a project administrator or end user.

8.5.7 Other components

- **The queue**
  - A central hub for passing messages between daemons. Usually implemented with RabbitMQ, but can be implemented with an AMQP message queue, such as Apache Qpid or ZeroMQ.

- **SQL database**
  - Stores most build-time and run-time states for a cloud infrastructure, including:
    - Available instance types
    - Instances in use
    - Available networks
    - Projects.
Theoretically, OpenStack Compute can support any database that SQL-Alchemy supports. Common databases are SQLite3 for test and development work, MySQL, and PostgreSQL. SQL-Alchemy is the Python SQL toolkit and Object Relational Mapper (ORM) that gives application developers the full power and flexibility of SQL. It provides a full suite of well known enterprise-level persistence patterns, designed for efficient and high-performing database access, adapted into a simple and Pythonic domain language.

8.5.8 Controller node

Three Compute service components are enabled on the controller node and one service component on the compute node.

```
oSBash@controller:~$ openstack compute service list
+----+------------------+------------+----------+---------+-------+----------------------------+
| ID | Binary           | Host       | Zone     | Status  | State | Updated At                 |
+----+------------------+------------+----------+---------+-------+----------------------------+
|  1 | nova-scheduler   | controller | internal | enabled | up    | 2017-09-24T12:21:29.000000  |
|  5 | nova-consoleauth | controller | internal | enabled | up    | 2017-09-24T12:21:25.000000  |
|  6 | nova-conductor   | controller | internal | enabled | up    | 2017-09-24T12:21:22.000000  |
|  8 | nova-compute     | compute1   | nova     | enabled | up    | 2017-09-24T12:21:27.000000  |
+----+------------------+------------+----------+---------+-------+----------------------------+
```

8.6 Networking service - Neutron

OpenStack Networking allows for the creation and attachment of interface devices managed by other OpenStack services to networks, the Virtual Networking Infrastructure (VNI) and how it accesses the Physical Networking Infrastructure (PNI). Plug-ins can be implemented to accommodate different networking equipment and software, providing flexibility to OpenStack architecture and deployment.

To do this Neutron provides object extractions to mimic its physical counterpart:

- Networks
- Subnets
- Routers
- Security Groups.

Neutron includes the following components:

- **neutron-server**
  - Accepts and routes API requests to the appropriate networking plug-in for action.

- **Networking plug-ins and agents**
  - Plug and unplug ports, create networks or subnets, and provide IP addressing. These plug-ins and agents differ depending on the vendor and technologies used in the particular cloud
  - Networking ships with plug-ins and agents for Cisco virtual and physical switches, NEC OpenFlow products, Open vSwitch (OvS), Linux bridging, and the VMware NSX product
  - The common agents are L3 (layer 3), Dynamic Host Configuration Protocol (DHCP), and a plug-in agent.
- **Messaging queue**
  - Used by most OpenStack Networking installations to route information between the *neutron-server* and various agents. It also acts as a database to store networking state for particular plug-ins.

*Neutron* mainly interacts with *Nova Compute* to provide networks and connectivity for its instances.

### 8.6.1 Controller node

List loaded extensions to verify successful launch of the neutron-server process.

```bash
osbash@controller:~$ openstack extension list --column "Name" --network
```

<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Subnetpools</td>
</tr>
<tr>
<td>Network IP Availability</td>
</tr>
<tr>
<td>Network Availability Zone</td>
</tr>
<tr>
<td>Auto Allocated Topology Services</td>
</tr>
<tr>
<td>Neutron L3 Configurable external gateway mode</td>
</tr>
<tr>
<td>Port Binding</td>
</tr>
<tr>
<td>agent</td>
</tr>
<tr>
<td>Subnet Allocation</td>
</tr>
<tr>
<td>L3 Agent Scheduler</td>
</tr>
<tr>
<td>Tag support</td>
</tr>
<tr>
<td>Neutron external network</td>
</tr>
<tr>
<td>Tag support for resources with standard attribute: trunk, policy,</td>
</tr>
<tr>
<td>security_group, floatingip</td>
</tr>
<tr>
<td>Neutron Service Flavors</td>
</tr>
<tr>
<td>Network MTU</td>
</tr>
<tr>
<td>Availability Zone</td>
</tr>
<tr>
<td>Quota management support</td>
</tr>
<tr>
<td>If-Match constraints based on revision_number</td>
</tr>
<tr>
<td>HA Router extension</td>
</tr>
<tr>
<td>Provider Network</td>
</tr>
<tr>
<td>Multi Provider Network</td>
</tr>
<tr>
<td>Quota details management support</td>
</tr>
<tr>
<td>Address scope</td>
</tr>
<tr>
<td>Neutron Extra Route</td>
</tr>
<tr>
<td>Network MTU (writable)</td>
</tr>
<tr>
<td>Subnet service types</td>
</tr>
<tr>
<td>Resource timestamps</td>
</tr>
<tr>
<td>Neutron Service Type Management</td>
</tr>
<tr>
<td>Router Flavor Extension</td>
</tr>
<tr>
<td>Port Security</td>
</tr>
<tr>
<td>Neutron Extra DHCP options</td>
</tr>
<tr>
<td>Resource revision numbers</td>
</tr>
<tr>
<td>Pagination support</td>
</tr>
<tr>
<td>Sorting support</td>
</tr>
<tr>
<td>security-group</td>
</tr>
<tr>
<td>DHCP Agent Scheduler</td>
</tr>
<tr>
<td>Router Availability Zone</td>
</tr>
<tr>
<td>RBAC Policies</td>
</tr>
<tr>
<td>Tag support for resources: subnet, subnetpool, port, router</td>
</tr>
<tr>
<td>standard-attr-description</td>
</tr>
<tr>
<td>Neutron L3 Router</td>
</tr>
<tr>
<td>Allowed Address Pairs</td>
</tr>
<tr>
<td>project_id field enabled</td>
</tr>
<tr>
<td>Distributed Virtual Router</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
</tr>
</tbody>
</table>

List agents to verify successful launch of the neutron agents. Four agents are shown on the controller node and one agent on the compute node.

```
osbash@controller:~$ openstack network agent list
```
8.6.2 Networking

The network configuration uses a provider (external) network that connects to the physical network infrastructure via layer-2 (bridging/switching). This network includes a DHCP server that provides IP addresses to instances.

The provider network uses 203.0.113.0/24 with a gateway on 203.0.113.1. The DHCP server assigns each instance a floating IP address from the range 203.0.113.101 - 203.0.113.250. All instances use 8.8.4.4 as a DNS resolver. It is worth noting that on the instance VM itself the floating IP address is not known. Neutron acts as a NAT router mapping the internal private IP address with the floating IP address.

```
oshbash@controller:~$ openstack network list --external
+--------------------------------------+-------------+--------------------------------------+
| ID                                   | Name        | Subnets                              |
+--------------------------------------+-------------+--------------------------------------+
| 7dff5340-fdfe-4c7b-ba43-17d60bfd0208 | selfservice | 11623486-cb6d-4295-88e1-89cc004b8f2   |
| a031d902-81fe-4bf3-933d-aec94296d1e0 | provider    | c87afd71-864f-47da-afc9-0087c52a13f9 |
+--------------------------------------+-------------+--------------------------------------+

oshbash@controller:~$ openstack subnet show provider
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| allocation_pools  | 203.0.113.101-203.0.113.200          |
| cidr              | 203.0.113.0/24                       |
| created_at        | 2017-09-24T14:31:54Z                 |
| description       |                                      |
| dns_nameservers   | 8.8.4.4                              |
| enable_dhcp       | True                                 |
| gateway_ip        | 203.0.113.1                          |
| host_routes       |                                      |
| id                | c87afd71-864f-47da-afc9-0087c52a13f9 |
| ip_version        | 4                                    |
| ipv6_address_mode | None                                 |
| ipv6_ra_mode      | None                                 |
| name              | provider                             |
| network_id        | a031d902-81fe-4bf3-933d-aec94296d1e0 |
| project_id        | 8209a2ba20634859baea07e34306deda     |
| revision_number   | 2                                    |
| segment_id        | None                                 |
| service_types     |                                      |
| subnetpool_id     | None                                 |
| updated_at        | 2017-09-24T14:31:54Z                 |
+-------------------+--------------------------------------+
```
8.6.3 Masquerade on virtualBox host

To enable VM instances to access the Internet it will be necessary to enable masquerading (Network Address Translation (NAT)) on the hypervisor host. The `nat_tables.sh` script in Appendix 1 will carry out that function. Create the script in `$OS_LAB` and make it executable. Run the script.

```
ada:~$ sudo $OS_LAB/nat_tables.sh
[sudo] password for alovelace: babbage
```

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

```
Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source          destination
  0     0 MASQUERADE all --  any  enp0s3  anywhere    anywhere

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source          destination
  0     0 ACCEPT  all --  enp0s3  vboxnet1  anywhere   anywhere    state RELATED,ESTABLISHED
  0     0 ACCEPT  all --  vboxnet1 enp0s3    anywhere   anywhere
```

8.7 Block Storage service - Cinder

*Cinder*, the OpenStack Block Storage service adds persistent storage to a VM. Block Storage provides an infrastructure for managing volumes, and interacts with OpenStack *Compute* to provide volumes for instances. The service also enables management of volume snapshots, and volume types.

The Block Storage service consists of the following components:

- **cinder-api**
  - Accepts API requests, and routes them to the cinder-volume for action.

- **cinder-volume**
  - Interacts directly with the Block Storage service, and processes such as the cinder-scheduler
  - Interacts with these processes through a message queue
  - Responds to read and write requests sent to the Block Storage service to maintain state
  - Interacts with a variety of storage providers through a driver architecture.

- **cinder-scheduler daemon**
  - Selects the optimal storage provider node on which to create the volume.

- **cinder-backup daemon**
  - Provides backing up volumes of any type to a backup storage provider
  - Interacts with a variety of storage providers through a driver architecture.

- **Messaging queue**
  - Routes information between the Block Storage processes.
8.7.1 Controller node

List service components to verify successful launch of each process. In this case the *cinder-scheduler* and *cinder-volume* should be active.

```
osbash@controller:~$ openstack volume service list
+------------------+--------------+------+---------+-------+----------------------------+
| Binary           | Host         | Zone | Status  | State | Updated_at                 |
+------------------+--------------+------+---------+-------+----------------------------+
| cinder-scheduler | controller   | nova | enabled | up    | 2017-09-24T13:19:22.000000 |
| cinder-volume    | compute1@lvm | nova | enabled | up    | 2017-09-24T13:19:18.000000 |
+------------------+--------------+------+---------+-------+----------------------------+
```

8.8 Orchestration service - Heat

Heat, the Orchestration service provides a template-based orchestration for describing a cloud application by running OpenStack API calls to generate running cloud applications. The software integrates other core components of OpenStack into a one-file template system. The templates allow for the creation of most OpenStack resource types such as instances, floating IPs, volumes, security groups, and Users. It also provides advanced functionality such as instance High Availability (HA), instance auto-scaling, and nested stacks. This enables OpenStack core projects to receive a larger user base.

The service is enabled for integration with the Orchestration service directly or through custom plug-ins.

The Orchestration service consists of the following components:

- **heat command-line client**
  - A CLI that communicates with the *heat-api* to run AWS CloudFormation APIs
  - End developers can directly use the Orchestration REST API.
- **heat-api component**
  - An OpenStack-native REST API that processes API requests by sending them to the heat-engine over Remote Procedure Call (RPC).
- **heat-api-cfn component**
  - An AWS Query API that is compatible with AWS CloudFormation
  - Processes API requests by sending them to the heat-engine over RPC.
- **heat-engine**
  - Orchestrates the launching of templates and provides events back to the API consumer.
osbash@controller:~$ openstack orchestration service list

OpenStack Training Laboratory
8.9 Instance flavour

VM instance profiles are called *flavours*. These flavours specify the virtual resource allocation profile which includes *processor*, *memory*, and *storage*. The smallest default flavour consumes 512 Megabytes (MB) memory per instance. For environments with compute nodes containing less than 4 Gigabytes (GB) memory, it is recommended creating the *m1.nano* flavour that only requires 64 MB per instance. This flavour is only suitable for use with the CirrOS image for testing purposes. Note: American spelling in command.

Typically the following *flavours* are established.

<table>
<thead>
<tr>
<th>Name</th>
<th>ID</th>
<th>vCPUs</th>
<th>RAM</th>
<th>Root Disk Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1.tiny</td>
<td>0</td>
<td>1</td>
<td>512 MB</td>
<td>1 GB</td>
</tr>
<tr>
<td>m1.small</td>
<td>1</td>
<td>1</td>
<td>2048 MB</td>
<td>20 GB</td>
</tr>
<tr>
<td>m1.medium</td>
<td>2</td>
<td>2</td>
<td>4096 MB</td>
<td>40 GB</td>
</tr>
<tr>
<td>m1.large</td>
<td>3</td>
<td>4</td>
<td>8192 MB</td>
<td>80 GB</td>
</tr>
<tr>
<td>M1.xlarge</td>
<td>4</td>
<td>8</td>
<td>16384 MB</td>
<td>160 GB</td>
</tr>
</tbody>
</table>

```
osbash@controller:~$ openstack flavor list
```

```
osbash@controller:~$ openstack flavor create --id 0 --vcpus 1 --ram 64 --disk 1 m1.nano
```

+----------------------------+---------+
| Field                      | Value   |
| +----------------------------+---------+
| OS-FLV-DISABLED:disabled   | False   |
| OS-FLV-EXT-DATA:ephemeral  | 0       |
| disk                       | 1       |
| id                         | 0       |
| name                       | m1.nano |
| os-flavor-access:is_public | True    |
| properties                 |         |
| ram                        | 64      |
| rxtx_factor                | 1.0     |
| swap                       |         |
| vcpus                      | 1       |
+----------------------------+---------+
osbash@controller:~$ openstack flavor create --id 1 --vcpus 1 --ram 2048 --disk 1 m1.small
+---------------------------------+----------+
| Field                            | Value    |
+---------------------------------+----------+
| OS-FLV-DISABLED:disabled         | False    |
| OS-FLV-EXT-DATA:ephemeral       | 0        |
| disk                            | 1        |
| id                              | 1        |
| name                            | m1.small |
| os-flavor-access:is_public      | True     |
| properties                      |          |
| ram                             | 2048     |
| rxtx_factor                     | 1.0      |
| swap                            |          |
| vcpus                           | 1        |

osbash@controller:~$ openstack flavor list
+----+----------+------+------+-----------+-------+-----------+
| ID | Name     | RAM | Disk | Ephemeral | VCPUs | Is Public |
+----+----------+------+------+-----------+-------+-----------+
| 0  | m1.nano  | 64  | 1    | 0         | 1     | True      |
| 1  | m1.small | 2048 | 1    | 0         | 1     | True      |
+----+----------+------+------+-----------+-------+-----------+
9. Deploying a VM instance

To deploy an instance the *compute* node has a running *KVM/QEMU* hypervisor. This hypervisor will spin up the VM. There are some other requirements. Where does the image come from?, the *Glance* service, security is provided from the *Nova* service, networking is provided by the *Neutron* service and storage from the *Cinder* service and finally the instance itself from the *Nova* service. The *Neutron* service will need a private network that will be reserved for that specific *Project* to run the instance on.

9.1 Deploying an Instance

To deploy an instance, the following steps will be followed:

1. Configure networking
2. Assign floating IP addresses
3. Define a security group in the cloud
4. Create an SSH key pair
5. Create a Glance image
6. Choose a flavour
7. The instance can be booted.
Networking is an essential part of the cloud environment. Therefore an SDN environment is created before starting to create instances. This SDN environment allows instances to connect to a private internal network, and also assign a floating IP address, so that they can be reached externally.

Illustration 8 shows two two instances: *Instance 1* and *Instance 2* with private IP addresses 192.168.10.21 and 192.168.10.22 respectfully. This network behaves like a network behind a NAT router. The IP addresses 192.168.10.21 and the 192.168.10.22 cannot be accessed directly from the external network. To allow access to the VM instances the SDN routing function has floating IP addresses, in this example from the 203.0.113.0/24 network, that can access the external network. The floating IP address is an IP address that is reserved for an instance, and is exposed at the external side of the SDN router. External traffic access the instance via its floating IP address.
9.3 Controller node

Enable the `demo-openrc` variables to apply the `demo User` credentials. This gives access to the user view commands. Users activity within `Projects` and the `admin` has no visibility of VM instances created by Users, this is because OpenStack is an orchestration tool and the infrastructure provider has no need to know what their customers orchestrate.

```bash
osbash@controller:$ . demo-openrc.sh
```

9.3.1 Generate a key pair

Cloud images typically support public key authentication instead of conventional password authentication. It is therefore necessary to add a public key to the `Compute` service that can be selected when launching the instance. Generate a key pair, the `. pem` file will be shared with Users who require access.

```bash
osbash@controller:$ openstack keypair create mykey > mykey.pem
osbash@controller:$ openstack keypair show mykey
```

```
+-------------+-------------------------------------------------+
| Field       | Value                                           |
+-------------+-------------------------------------------------+
| created_at  | 2017-09-25T14:18:30.000000                      |
| deleted     | False                                           |
| id          | 2                                               |
| name        | mykey                                           |
| updated_at  | None                                            |
| user_id     | d3182eca0e80481ea6ab018ab7de9bb5                |
+-------------+-------------------------------------------------+
```

```bash
osbash@controller:$ cat mykey.pem
```

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAyi3HFYVllk6wXMUK7jIPx4ZLXXCMW4mBT8y3CrjChpOjKo
ghlK0WRUCn1MROaEskKz4d9S8F0qE10gKxns1fFpWxExyK2zcxzBOjE4Xv9t9
UyvWGbHyZbcFhJ95smW+1P3YKoWPDtdWx0l7ybyB6w9s5gJ5uacvBqgo+AAA0
40c6KxKvwUA2p/PP56FR/P8VUyVmjwFK3vSN5048TEcmYpmc424j12DFP3gPM
yRkBCq4Q5QThuyUXWxXh3cnuCn64nHUN4vCldjixB0dQuiE38yPKeFQhWmLyFl
xtFfJzvuler/evVg01t+CaN46HwzrpgMSHA1Dped9QlDAQABAo1BAG7gTssp4FEnKtL
VAXMFfGp32j2JbGUn/s1ge+1gFAL10odGR7dep/KVmnHyGhD93dKbc1JZhRbqPTc
Tr7cCN0P+yaH1E8qCvtesP21d4nGVYqz2KRU42KUV17TS31sHts5EHhKg786nLDL
Pld/Fkd4aidRnxYYZZq/QU6eUioPo5ys1JhldL5+hbCqEvXUlkRWBri8r7TUq9j
z+GRq2zwq1pcwheiLu/pdRgxwFSpVDZ21M6+B6/ru/vROkJIVsu7xK1ex1gMF+r6
GxWwV/vQJ9h9w7CmP207JGPTzFpovjdAM181pEvKipiNGMC9t5pFm01J7NWbNpO
fSkn5a5CQyEA83e7wvpekKlCFNJVWwLgaGwKCAQ/Vk1gNd1KjutJ64vmrR9y0s0r
f/60YJ+w6WPpNsbAAxqT7+Gpeya2B8xq1VgNXO40LIP4KtvQv6XaekswuLYXworA
OWN3L0V5kCMKx+1b90sPQ2FSt8cPbJ+m1130N1D1BqkneaVXHerCqYEA1JgLg
rv6Tm40oN1Fmum8N99he8jve4SyC7JUmgzb+z3jmnJnngDhNtxRtT9g9kmvTuu
zkQB6pgoxNvy78hLut4ta7AkoKIApiE1bnchQV1lpfxsf+kkcZNUQxZ2dWp1wJ
7L+5n+VwicaN3ls1S/2wC6Ug7dGME07kCcMcgyEA43mJz0JceHpsvr6SMKFQ/4
x730JYHWSF2T9BtvO7rO9Va030q99q8G0q9390q30E1i90uzsae1IdJ3r38Uowvi
8JHNYC2VYAYa8681IdCjur|zFreUrCyvOGuEuFd2DB20JknMLTJ9m9sr41on38925PMs
OVnCDSuegYhKqXDPvU0BhMCGYAYEBCF28PA4888b8c8yQa+7Q10ystHt6+ZkU9w9j
kHEWQu1G1S2Fub8Ks9536GAD1J7F7HoCr+LGuTFMmJnYdck4s7u/G48Pf7QoYtJ
Gz+flvTV3osAcWUjbr+90B82PhKxwos1Pnup7M6ShCC4m+8TJ3v811761joJu7s7m
Pr3U6kpQkgYXCM21QbMcPerc09hGxYveqO118k7F5FMxXnJsiD71TAMFUQs806bLUl
MYjGmYVQcYJw4K2Emh13p8wFpEg5qSty2p6sG8cD96cYVQmks4K9l40Cmqy2736b
K5DhOa779q7pxYTuUSQ43hClppqy5oDa2zq14KCrhYNSAnHdqUy+YAAA
-----END RSA PRIVATE KEY-----
```
Set the permissions of the .pem file so that only you can read and write to it, run the following command.

```
osbash@controller:~$ chmod 600 mykey.pem
```

The key is now added.

Download the .pem file to the computer that will connect to the VM instance. In this case to the host computer at 192.168.10.2.

```
osbash@controller:~$ sftp alovelace@192.168.10.2
alovelace@192.168.10.2's password: babbage
Connected to 192.168.10.2.
sftp> cd OpenStack-lab
sftp> put mykey.pem
Uploading mykey.pem to /home/alovelace/OpenStack-lab/mykey.pem
mykey.pem
100% 1680  1.6KB/s  00:00
sftp> exit
```
9.3.2 Security group

There exists by default security group called *default*.

```bash
osbash@controller:~$ openstack security group list
```

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Description</th>
<th>Project</th>
</tr>
</thead>
<tbody>
<tr>
<td>c62678-8b51-42b5-824c-6c9babc152</td>
<td>default</td>
<td>default security group</td>
<td>9a10483c14d680e0e294ac545e2</td>
</tr>
</tbody>
</table>
By default this security group `default` is quite restricted. For the purpose of this exercise permit both SSH and Internet Control Message Protocol (ICMP).

```bash
osbash@controller:~$ openstack security group rule create --proto icmp default
+-------------------------------+--------------------------------------+
| Field                         | Value                                |
+-------------------------------+--------------------------------------+
| created_at                   | 2017-09-25T15:08:39Z                 |
| description                  |                                      |
| direction                    | ingress                             |
| ether_type                   | IPv4                                |
| id                           | 01b98dc9-5a07-4556-8eee-ee0ec35b4eed |
| name                         | None                                |
| port_range_max               | None                                |
| port_range_min               | None                                |
| project_id                   | 9a10148d3c414d61800ee2946ac545ea    |
| protocol                     | icmp                                |
| remote_group_id              | None                                |
| remote_ip_prefix             | 0.0.0.0/0                           |
| revision_number              | 0                                   |
| security_group_id            | c6d26784-b591-42b5-8624-6c29bebcb152 |
| updated_at                   | 2017-09-25T15:08:39Z                 |
+-------------------------------+--------------------------------------+
```

```bash
osbash@controller:~$ openstack security group rule create --proto tcp --dst-port 22 default
+-------------------------------+--------------------------------------+
| Field                         | Value                                |
+-------------------------------+--------------------------------------+
| created_at                   | 2017-09-25T15:08:57Z                 |
| description                  |                                      |
| direction                    | ingress                             |
| ether_type                   | IPv4                                |
| id                           | dfe7f03b-4149-4c58-8fd9-a0983d3855f5  |
| name                         | None                                |
| port_range_max               | 22                                  |
| port_range_min               | 22                                  |
| project_id                   | 9a10148d3c414d61800ee2946ac545ea    |
| protocol                     | tcp                                 |
| remote_group_id              | None                                |
| remote_ip_prefix             | 0.0.0.0/0                           |
| revision_number              | 0                                   |
| security_group_id            | c6d26784-b591-42b5-8624-6c29bebcb152 |
| updated_at                   | 2017-09-25T15:08:57Z                 |
+-------------------------------+--------------------------------------+
```
```bash
osbash@controller:~$ openstack security group rule list default
```

<table>
<thead>
<tr>
<th>ID</th>
<th>IP Protocol</th>
<th>IP Range</th>
<th>Port Range</th>
<th>Remote Security Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>41086641</td>
<td>tcp</td>
<td>0.0.0.0/0</td>
<td>0-0</td>
<td>None</td>
</tr>
<tr>
<td>46707026</td>
<td>tcp</td>
<td>0.0.0.0/0</td>
<td>22-22</td>
<td>None</td>
</tr>
<tr>
<td>52864585</td>
<td>tcp</td>
<td>0.0.0.0/0</td>
<td>56-60</td>
<td>c6d27974-8651-4205-8624-6c23bebc152</td>
</tr>
<tr>
<td>59873588</td>
<td>tcp</td>
<td>0.0.0.0/0</td>
<td>56-60</td>
<td>c6d27974-8651-4205-8624-6c23bebc152</td>
</tr>
<tr>
<td>66882155</td>
<td>tcp</td>
<td>0.0.0.0/0</td>
<td>56-60</td>
<td>c6d27974-8651-4205-8624-6c23bebc152</td>
</tr>
<tr>
<td>73890721</td>
<td>tcp</td>
<td>0.0.0.0/0</td>
<td>56-60</td>
<td>c6d27974-8651-4205-8624-6c23bebc152</td>
</tr>
<tr>
<td>80900182</td>
<td>tcp</td>
<td>0.0.0.0/0</td>
<td>56-60</td>
<td>c6d27974-8651-4205-8624-6c23bebc152</td>
</tr>
<tr>
<td>87911238</td>
<td>tcp</td>
<td>0.0.0.0/0</td>
<td>56-60</td>
<td>c6d27974-8651-4205-8624-6c23bebc152</td>
</tr>
<tr>
<td>94922583</td>
<td>tcp</td>
<td>0.0.0.0/0</td>
<td>56-60</td>
<td>c6d27974-8651-4205-8624-6c23bebc152</td>
</tr>
<tr>
<td>101934088</td>
<td>tcp</td>
<td>0.0.0.0/0</td>
<td>56-60</td>
<td>c6d27974-8651-4205-8624-6c23bebc152</td>
</tr>
<tr>
<td>108945292</td>
<td>tcp</td>
<td>0.0.0.0/0</td>
<td>56-60</td>
<td>c6d27974-8651-4205-8624-6c23bebc152</td>
</tr>
<tr>
<td>115956422</td>
<td>tcp</td>
<td>0.0.0.0/0</td>
<td>56-60</td>
<td>c6d27974-8651-4205-8624-6c23bebc152</td>
</tr>
</tbody>
</table>

No rules are currently defined.
9.3.3 Create volume

Create a 1 GB logical volume that can be attached to a VM instance later. **Cinder** uses LVM in GNU/Linux. LVM manages disk drives and similar mass-storage devices. Volume refers to a disk drive or partition of a disk drive. It was written in 1998 by Heinz Mauelshagen, who based its design on that of the LVM in HP-UX. LVM can be considered as a thin software layer on top of the hard disks and partitions, which creates an abstraction of continuity and ease-of-use for managing hard drive replacement, re-partitioning, and backup. Cinder creates the volume on the **compute node**.

```
osbash@controller:~$ openstack volume create --size 1 1GB-vol
```

| Field               | Value                                |
|---------------------+--------------------------------------|
+---------------------+--------------------------------------|
| attachments         | []                                   |
| availability_zone   | nova                                 |
| bootable            | false                                |
| consistencygroup_id | None                                 |
| created_at          | 2017-09-24T15:09:25.737145            |
| description         | None                                 |
| encrypted           | False                                |
| id                  | 10c8ca56-e831-4ada-9a6c-a2ee7b3a03ed  |
| multiattach         | False                                |
| name                | 1GB-vol                              |
| properties          |                                      |
| replication_status  | None                                 |
| size                | 1                                    |
| snapshot_id         | None                                 |
| source_volid        | None                                 |
| status              | creating                             |
| type                | None                                 |
| updated_at          | None                                 |
| user_id             | 3fc36085c8fd4472bf507d03fa50dcd2     |
+---------------------+--------------------------------------|

```
osbash@controller:~$ openstack volume list
```

<table>
<thead>
<tr>
<th>ID</th>
<th>Display Name</th>
<th>Status</th>
<th>Size</th>
<th>Attached to</th>
</tr>
</thead>
<tbody>
<tr>
<td>10c8ca56-e831-4ada-9a6c-a2ee7b3a03ed</td>
<td>1GB-vol</td>
<td>available</td>
<td>1G</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

9.3.3.1 Compute node

Looking at the block devices on the **compute node**. Cinder has a volume on the `/dev/sdb`.

```
osbash@compute1:~$ lsblk
NAME                   MAJ:MIN  RM SIZE RO TYPE MOUNTPOINT
sda                    8:0       0 9.8G  0 disk
 | -sda1                 8:1       0 9.3G  0 part /
 | -sda2                 8:2       0  1G  0 part
 | -sda5                 8:5       0 512M  0 part [SWAP]
| `cinder--volumes-volume--10c8ca56--e831--4ada--9a6c--a2ee7b3a03ed 252i:0 0 1G 0 lvm
| `-cinder--volumes-volume--10c8ca56--e831--4ada--9a6c--a2ee7b3a03ed 252i:0 0 1G 0 lvm
```

```
osbash@compute1:~$ lsblk
NAME                   MAJ:MIN  RM SIZE RO TYPE MOUNTPOINT
sdb                    8:16      0 200G  0 disk
 | `cinder--volumes-volume--10c8ca56--e831--4ada--9a6c--a2ee7b3a03ed 252i:0 0 1G 0 lvm
| `-cinder--volumes-volume--10c8ca56--e831--4ada--9a6c--a2ee7b3a03ed 252i:0 0 1G 0 lvm
```

92 OpenStack Training Laboratory

---

OpenStack Pike

02 Oct 2017
Using the LVM display commands it is possible to see the newly created volume on the compute node. Firstly the `pvdisplay` command shows the Physical Volume, the `vgdisplay` command the Volume Group and finally the `lvdisplay` command the Logical Volume. On a production system it is possible to have multiple Logical Volumes in a Volume Group.

```
osbash@compute1:~$ sudo pvdisplay
--- Physical volume ---
PV Name               /dev/sdb
VG Name               cinder-volumes
PV Size               200.00 GiB / not usable 4.00 MiB
Allocatable           yes
PE Size               4.00 MiB
Total PE              51199
Free PE               50943
Allocated PE          256
PV UUID               5ULpY2-FTXz-vxMs-0bvp-0wT8-0pu1-DcMU1O
```

```
osbash@compute1:~$ sudo vgdisplay
--- Volume group ---
VG Name               cinder-volumes
System ID
Format                lvm2
Metadata Areas        1
Metadata Sequence No  4
VG Access             read/write
VG Status             resizable
MAX LV                0
Cur LV                1
Open LV               0
Max PV                0
Cur PV                1
Act PV                1
VG Size               200.00 GiB
PE Size               4.00 MiB
Total PE              51199
Alloc PE / Size       256 / 1.00 GiB
Free  PE / Size       50943 / 199.00 GiB
VG UUID               WRJWFo-yWF4-Q92e-J8KV-r8a7-wV3v-Y90T05
```

```
osbash@compute1:~$ sudo lvdisplay
--- Logical volume ---
LV Path                /dev/cinder-volumes/volume-10c8ca56-e831-4ada-9a6c-a2ee7b3a03ed
LV Name                volume-10c8ca56-e831-4ada-9a6c-a2ee7b3a03ed
VG Name                cinder-volumes
LV UUID                rG10wI-W4aB-kMuE-9BY0-XSTi-oNkj-t7tfp2
LV Write Access        read/write
LV Creation host, time compute1, 2017-09-24 15:09:29 +0000
LV Status              available
# open                 0
LV Size                1.00 GiB
Current LE             256
Segments               1
Allocation             inherit
Read ahead sectors     auto
- currently set to     256
Block device           252:0
```
9.3.4 Launch a CirrOS instance

As the demo user check the flavours available, in this case m1.nano will be used. Get the ID of the provider network and the available security group and the name of the OS image that is required.

```
osbash@controller:~$ . demo-openrc.sh

osbash@controller:~$ openstack flavor list
+----+-----------+------+------+-----------+-------+-----------+
| ID | Name      |  RAM | Disk | Ephemeral | VCPUs | Is Public |
+----+-----------+------+------+-----------+-------+-----------+
| 0  | m1.nano   |   64 |    1 |         0 |     1 | True      |
| 1  | m1.small  | 2048 |    1 |         0 |     1 | True      |
+----+-----------+------+------+-----------+-------+-----------+

osbash@controller:~$ openstack network list
+--------------------------------------+-------------+--------------------------------------+
| ID                                   | Name        | Subnets                              |
+--------------------------------------+-------------+--------------------------------------+
| 7dff5340-fdfe-4c7b-ba43-17d60b0d0208 | selfservice | 11623b6e-cb6d-4295-88e1-89ccd004b8f2 |
| a031d902-81fe-4bf3-933d-aec94296d1e0 | provider    | c87af6f1-854f-47da-aec9-0087c52a13f3 |
+--------------------------------------+-------------+--------------------------------------+

osbash@controller:~$ NIC=$(openstack network list | grep provider | awk '{print $2}')

osbash@controller:~$ echo $NIC
a031d902-81fe-4bf3-933d-aec94296d1e0
```
osbash@controller:~$ openstack security group list

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>c6d26fb4-b591-4db5-8624-6c39ebc152</td>
<td>default</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

osbash@controller:~$ openstack image list

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>79d5847b-bfd4-47e8-badfc-ec219687d4e</td>
<td>cirros</td>
<td>active</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Now that the preparatory work is complete, launch the instance.

```bash
osbash@controller:~$ openstack server create --flavor m1.nano --image cirros --nic net-id=$NIC --security-group default --key-name mykey cirrOS-test
```

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS-DCF:diskConfig</td>
<td>MANUAL</td>
</tr>
<tr>
<td>OS-EXT-AZ:availability_zone</td>
<td></td>
</tr>
<tr>
<td>OS-EXT-STS:power_state</td>
<td>NOSTATE</td>
</tr>
<tr>
<td>OS-EXT-STS:task_state</td>
<td>scheduling</td>
</tr>
<tr>
<td>OS-EXT-STS:vm_state</td>
<td>building</td>
</tr>
<tr>
<td>OS-SRV-USG:launched_at</td>
<td>None</td>
</tr>
<tr>
<td>OS-SRV-USG:terminated_at</td>
<td>None</td>
</tr>
<tr>
<td>accessIPv4</td>
<td></td>
</tr>
<tr>
<td>accessIPv6</td>
<td></td>
</tr>
<tr>
<td>addresses</td>
<td></td>
</tr>
<tr>
<td>adminPass</td>
<td>CLmaDgtvoUP9</td>
</tr>
<tr>
<td>config_drive</td>
<td></td>
</tr>
<tr>
<td>created</td>
<td>2017-09-25T14:19:05Z</td>
</tr>
<tr>
<td>flavor</td>
<td>m1.nano (0)</td>
</tr>
<tr>
<td>hostId</td>
<td></td>
</tr>
<tr>
<td>id</td>
<td>f1b1e3a6-076a-4cfe-8411-cc8f4987b8be</td>
</tr>
<tr>
<td>image</td>
<td>cirros (fae8b59c-6193-4b34-bf8c-72eb60a73e0a)</td>
</tr>
<tr>
<td>key_name</td>
<td>mykey</td>
</tr>
<tr>
<td>name</td>
<td>cirrOS-test</td>
</tr>
<tr>
<td>progress</td>
<td>0</td>
</tr>
<tr>
<td>project_id</td>
<td>9a10148d3c414d61800ee2946ac545ea</td>
</tr>
<tr>
<td>properties</td>
<td></td>
</tr>
<tr>
<td>security_groups</td>
<td>name='c6d26784-b591-42b5-8624-6c29bebbbc152'</td>
</tr>
<tr>
<td>status</td>
<td>BUILD</td>
</tr>
<tr>
<td>updated</td>
<td>2017-09-25T14:19:05Z</td>
</tr>
<tr>
<td>user_id</td>
<td>d3182e9abe90481ea6ab018ab7de9bb5</td>
</tr>
<tr>
<td>volumes_attached</td>
<td></td>
</tr>
</tbody>
</table>

Check the running instance.

```bash
osbash@controller:~$ openstack server list
```

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Status</th>
<th>Networks</th>
<th>Image</th>
<th>Flavor</th>
</tr>
</thead>
<tbody>
<tr>
<td>f1b1e3a6-076a-4cfe-8411-cc8f4987b8be</td>
<td>cirrOS-test</td>
<td>ACTIVE</td>
<td>provider=203.0.113.102</td>
<td>cirros</td>
<td>m1.nano</td>
</tr>
</tbody>
</table>
### 9.3.5 Attach the volume

Attach the volume created earlier to the instance VM.

```bash
osbash@controller:~$ openstack server add volume cirrOS-test 1GB-vol
```

Confirm the attachment.

```bash
osbash@controller:~$ openstack volume list
```

```
+---------------------------------+------------------+
| ID                              | Display Name     |
+---------------------------------+------------------+
| 10:8d4e-8e34-4da2-96c5-2ee7d38a34d | 1GB-vol          |
+---------------------------------+------------------+
  Status                          |
  In-use                          |
  Size                            |
  1 GB                            |
  Attached to cirrOS-test on /dev/vdb|
+---------------------------------+------------------+
```

9.3.6 Connect to the new instance

Obtain a VNC session URL for the instance and access it from a web browser.

```
osbash@controller:~$ openstack console url show cirrOS-test
```

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>novnc</td>
</tr>
<tr>
<td>url</td>
<td><a href="http://10.0.0.11:6080/vnc_auto.html?token=1f972966-e631-4a64-8c6f-ed4f3b87c657">http://10.0.0.11:6080/vnc_auto.html?token=1f972966-e631-4a64-8c6f-ed4f3b87c657</a></td>
</tr>
</tbody>
</table>

Taking the URL given open a Virtual Console to the new instance.

![Virtual Console](illustration.png)

Illustration 9: Virtual Console
Alternatively SSH to the new instance.

```
ada:~$ ssh cirros@203.0.113.108
The authenticity of host '203.0.113.108 (203.0.113.108)' can't be established.
RSA key fingerprint is
SHA256:RTl11u32pYf11QeGj10iDLJBaE2t8e2p0u5vmR1wI6A.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '203.0.113.108' (RSA) to the list of known hosts.
cirros@203.0.113.108's password: cubswin:
```

```
$ cat /etc/os-release
NAME=Buildroot
VERSION=2012.05-dirty
ID=buildroot
VERSION_ID=2012.05
PRETTY_NAME="Buildroot 2012.05"
```

It is also possible to SSH using the `mykey.pem` key file instead of a password. This is actually the more typical method for accessing VM instances in the cloud. Note that no password is required in this case.

```
ada:~$ ssh -i mykey.pem cirros@203.0.113.108
```

```
$ cat /etc/os-release
NAME=Buildroot
VERSION=2012.05
ID=buildroot
VERSION_ID=2012.05
PRETTY_NAME="Buildroot 2012.05"
```

Test network connectivity.

### 9.3.7 Provider network gateway

```
$ ping -c1 203.0.113.1
PING 203.0.113.1 (203.0.113.1): 56 data bytes
64 bytes from 203.0.113.1: seq=0 ttl=64 time=2.374 ms

--- 203.0.113.1 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 2.374/2.374/2.374 ms
```
9.3.8 Host public interface

$ ping -c1 192.168.10.2
PING 192.168.91.100 (192.168.91.100): 56 data bytes
64 bytes from 192.168.91.100: seq=0 ttl=64 time=0.582 ms

--- 192.168.91.100 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.582/0.582/0.582 ms

9.3.9 IP Address on the public Internet

$ ping -c1 8.8.8.8
PING 8.8.8.8 (8.8.8.8): 56 data bytes
64 bytes from 8.8.8.8: seq=0 ttl=56 time=376.109 ms

--- 8.8.8.8 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 376.109/376.109/376.109 ms

9.4 Review the instance

Looking at the server instance launched on the controller node and the processes executed, it can be seen that the Universally Unique Identifier (UUID) in the process matches the ID of the server.

9.4.1 Controller node

osbash@controller:~$ . demo-openrc.sh
osbash@controller:~$ openstack server list

+--------------------------------------+-------------+--------+------------------------+--------+---------+
| ID                                   | Name        | Status | Networks               | Image  | Flavor  |
+--------------------------------------+-------------+--------+------------------------+--------+---------+
| f1b1e3a6-076a-4cfe-8411-cc8f4987b8be | cirrOS-test | ACTIVE | provider=203.0.113.102 | cirros | m1.nano |
+--------------------------------------+-------------+--------+------------------------+--------+---------+
9.4.2  Compute node

On the compute node it is possible to use the `virsh` tool to monitor the QEMU VM instances. `virsh` uses the `libvirt` C toolkit to interact with the virtualisation capabilities of GNU/Linux and while it supports many hypervisors like Xen, KVM, LXC, OpenVZ, VirtualBox and VMware ESX, it is its support for QEMU that is of interest here.

Get the domain ID of the instance from the QEMU hypervisor perspective.

```
osbash@compute1:~$ virsh list
Id   Name                           State
----------------------------------------------------
1    instance-00000001              running
```

On the compute node it is possible to use the `virsh` tool to monitor the QEMU VM instances.
With the domain ID use the `dominfo` and `domstats` commands to find out about the instance.

```
$ virsh dominfo instance-00000001
Id:             1
Name:           instance-00000001
UUID:           f1b1e3a6-076a-4cfe-8411-cc8f4987b8be
OS Type:        hvm
State:          running
CPU(s):         1
CPU time:       26.4s
Max memory:     65536 KiB
Used memory:    65536 KiB
Persistent:     yes
Autostart:      disable
Managed save:   no
Security model: apparmor
Security DOI:   0
Security label: libvirt-f1b1e3a6-076a-4cfe-8411-cc8f4987b8be (enforcing)
```

```
$ virsh domstats instance-00000001
Domain: 'instance-00000001'
  state.state=1
  state.reason=5
  cpu.time=2684845373
  cpu.user=18120000000
  cpu.system=4970000000
  balloon.current=65536
  balloon.maximum=65536
  balloon.last-update=0
  balloon.rss=192672
  vcpu.current=1
  vcpu.maximum=1
  net.count=1
    net.0.name=tap200bf068-19
    net.0.rx.bytes=46959
    net.0.rx.pkts=527
    net.0.rx.errs=0
    net.0.rx.drop=0
    net.0.tx.bytes=21708
    net.0.tx.pkts=192
    net.0.tx.errs=0
    net.0.tx.drop=0
  block.count=1
    block.0.name=vda
    block.0.path=/var/lib/nova/instances/f1b1e3a6-076a-4cfe-8411-cc8f4987b8be/disk
    block.0.rd.reqs=900
    block.0.rd.bytes=20292608
    block.0.rd.times=2835638911
    block.0.wr.reqs=148
    block.0.wr.bytes=415744
    block.0.wr.times=2753645734
    block.0.fl.reqs=34
    block.0.fl.times=1098870909
    block.0.allocation=2424832
    block.0.capacity=1073741824
    block.0.physical=2367488
```
9.5  Configure a volume in the new instance

Connect to the new instance and find the name of the attached volume.

```bash
ada:~$ ssh -i mykey.pem cirros@203.0.113.108
lsblk
NAME    MAJ:MIN  RM  SIZE RO TYPE  MOUNTPOINT
vda      253:0    0   1G  0 disk
  -vda1   253:1    0  1011.9M  0 part /
vdb      253:16   0   1G  0 disk
```

Build a partition `/dev/vdb1` on the volume `/dev/vdb`.

```bash
$ sudo fdisk /dev/vdb
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel
Building a new DOS disklabel with disk identifier 0x4b4c2ca7.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won’t be recoverable.

Command (m for help): n
Partition type:
  p   primary (0 primary, 0 extended, 4 free)
  e   extended
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-2097151, default 2048):
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-2097151, default 2097151):
Using default value 2097151

Command (m for help): w
The partition table has been altered!
Calling ioctl() to re-read partition table.
Syncing disks.

Command (m for help): p
Disk /dev/vdb: 1073 MB, 1073741824 bytes
9 heads, 8 sectors/track, 29127 cylinders, total 2097152 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x4d9ce5ea

Device Boot Start   End   Blocks   Id  System
/dev/vdb1   2048  2097151  1047552 83  Linux

Command (m for help): q
Build a new Linux ext4 filesystem on the partition /dev/vdb1.

```
$ sudo mkfs.ext4 /dev/vdb1
mke2fs 1.42.2 (27-Mar-2012)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
65536 inodes, 261888 blocks
13094 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=268435456
8 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376

Allocating group tables: done
Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done
```

Create a /mnt/1GB-vol mount point for the new volume.

```
$ sudo mkdir /mnt/1GB-vol
```

Mount the volume dev/vdb1 on the mount point /mnt/1GB-vol.

```
$ sudo mount -t ext4 /dev/vdb1 /mnt/1GB-vol/
```

```
$ df -h
Filesystem                Size      Used Available Use% Mounted on
/dev                     21.3M         0     21.3M   0% /dev
/dev/vda1                23.2M     18.0M      4.0M  82% /
tmpfs                    24.8M         0     24.8M   0% /dev/shm
tmpfs                   200.0K     72.0K    128.0K  36% /run
/dev/vdb1              1006.9M     17.3M    938.5M   2% /mnt/1GB-vol
```
10. **Scripting the building and launching new instance**

To speed up the testing process the script from Appendix 3 can be used to build and launch a new instance. However if this document is being followed it will be first necessary to clean the nodes of existing work. The nodes can be returned to their original state with the script from Appendix 2. Install the `clean_nodes.sh` script in `$OS_LAB` and make it executable.

```
ada:~$ chmod +x $OS_LAB/clean_nodes.sh
```

### 10.1 Cleaning the nodes

```
ada:~$ $OS_LAB/clean_nodes.sh
```

Restoring nodes to clean state

Powering off both nodes

```
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
```

Waiting for nodes to power down completely

```
. . . . . . . . . .
```

Returning Controller node to snapshot 'public_private_networks'

```
Restoring snapshot c518b257-26c4-433e-8859-f1133e9f4b4d
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
```

Returning Controller node to snapshot 'cinder-volume_installed'

```
Restoring snapshot e852b306-1e42-4408-b36d-76bba90e2aeb
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
```

Restarting nodes

```
Waiting for VM "controller" to power on...
VM "controller" has been successfully started.
Waiting for VM "compute1" to power on...
VM "compute1" has been successfully started.
```

Clean running nodes

```
"controller" {a7878c03-d397-4ca2-bc1c-1cd05570d42d}
"compute1" {ecdba7db-6ce3-4e3a-8cd6-444c4983771}
```

### 10.2 Launch new instance

Connect to the `controller` node and install the `instance_launch.sh` script from Appendix 3 in the home directory. Change its permissions to make it executable.

```
osbash@controller:~$ chmod +x ~/instance_launch.sh
```

Take on the demo User credentials.

```
osbash@controller:~$ . demo-openrc.sh
```
Now launch new instance script.

```
osbash@controller:~$ ~/instance_launch.sh
```

admin-openrc script created

demo-openrc script created

Setting admin-openrc variables

Creating flavour m1.nano

Setting demo-openrc variables

Creating keypair mykey and ~/mykey.pem file

Restricting ~/mykey.pem access rights

Adding port 22 (SSH) and ICMP to default security group

```
<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>created_at</td>
<td>2016-12-29T12:23:30Z</td>
</tr>
<tr>
<td>description</td>
<td></td>
</tr>
<tr>
<td>direction</td>
<td>ingress</td>
</tr>
<tr>
<td>ethertype</td>
<td>IPv4</td>
</tr>
<tr>
<td>headers</td>
<td></td>
</tr>
<tr>
<td>id</td>
<td>500d708d-f061-431c-8e75-976be6e12874</td>
</tr>
<tr>
<td>port_range_max</td>
<td>22</td>
</tr>
<tr>
<td>port_range_min</td>
<td>22</td>
</tr>
<tr>
<td>project_id</td>
<td>8be62de81f7db486486fe11e2bd961301</td>
</tr>
<tr>
<td>project_id</td>
<td>8be62de81f7db486486fe11e2bd961301</td>
</tr>
<tr>
<td>protocol</td>
<td>tcp</td>
</tr>
<tr>
<td>remote_group_id</td>
<td>None</td>
</tr>
<tr>
<td>remote_ip_prefix</td>
<td>0.0.0.0/0</td>
</tr>
<tr>
<td>revision_number</td>
<td>1</td>
</tr>
<tr>
<td>security_group_id</td>
<td>7d45b7e8-4974-422a-943e-8b5277f659f4</td>
</tr>
<tr>
<td>updated_at</td>
<td>2016-12-29T12:23:30Z</td>
</tr>
</tbody>
</table>
```

Extracting provider network UUID: 1960a5c6-77eb-47b7-855e-e3a7bf86f183

Creating and launching instance cirrOS-test with:

```
Flavour: m1.nano
Image: cirros
Network UUID=1960a5c6-77eb-47b7-855e-e3a7bf86f183
Security group: default
Key name: mykey
```
Waiting for instance cirrOS-test to become ACTIVE

Creating volume 1GB-vol

Adding volume 1GB-vol to VM instance cirrOS-test
10.2.1 Confirm VM instance

```bash
osbash@controller:~$ openstack server list
```

```
+--------------------------------------+-------------+--------+------------------------+------------+
| ID                                   | Name        | Status | Networks               | Image Name |
+--------------------------------------+-------------+--------+------------------------+------------+
| 8affc840-ca6d-4084-a776-9858bc12981d | cirrOS-test | ACTIVE | provider=203.0.113.107 | cirros     |
+--------------------------------------+-------------+--------+------------------------+------------+
```
11. **Download and build a second Ubuntu image**

As practice build another VM. To do so with the training lab requires some adjustment to the `compute` node to accommodate a larger image.

11.1 **Compute node – KVM/QEMU**

On KVM/QEMU change the `compute` node memory and CPUs (Assuming the hardware can accommodate the change).

Confirm the current vCPU and memory available at the `compute` node.

```
ada:~$ ssh osbash@192.168.122.140
osbash@192.168.122.140's password: osbash
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-57-generic x86_64)

* Documentation: https://help.ubuntu.com
* Management:     https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage
Last login: Sat May  6 05:34:27 2017 from 192.168.122.1

osbash@compute1:~$ lscpu | grep '^CPU(s)
CPU(s):                1

osbash@compute1:~$ cat /proc/meminfo | grep MemTotal
MemTotal:        1016164 kB
```

There is 1 vCPU and 1 GB of memory.

Before working on the memory or the CPUs on a KVM/QEMU VM, shutdown the VM domain.

```
virsh # shutdown compute1
Domain compute1 is being shutdown

virsh # list --all
Id    Name                           State
----------------------------------------------------
65    controller                     running
-     compute1                       shut off
```

Change maximum memory limit and then the memory allocation. The first sets the limit for memory on this VM domain. It is then possible to dynamically modify the VM domain memory up to the max limit.

```
virsh # setmaxmem compute1 6G --config
virsh # setmem compute1 4G --config
```
Confirm these changes. (Note: using command from host shell as there is no `grep` within the `virsh` # shell).

```
ada:~$ virsh dumpxml compute1 | grep 'memory'
<memory unit='KiB'>6291456</memory>
```

```
ada:~$ virsh dumpxml compute1 | grep 'currentMemory'
<currentMemory unit='KiB'>4194304</currentMemory>
```

Before changing the number of vCPUs, confirm the number of CPUs on the host system. Obviously it is not possible to use this number as the host’s own requirements must be catered for.

```
virsh # maxvcpus
16
```

Edit the eXtensible Markup Language (XML) file for the VM domain to change the `vcpu placement` to 4. This will make 4 vCPU available to the VM domain.

```
virsh # edit compute1
...
<vcpu placement='static'>4</vcpu>
...
```

Apply the changes to the XML file. (Note: as `sudo` is required (the XML file is owned by `root`) the full form command is necessary). Confirm the vCPUs for the VM domain.

```
ada:~$ sudo virsh create /etc/libvirt/qemu/compute1.xml
Domain compute1 created from /etc/libvirt/qemu/compute1.xml
```

```
ada:~$ virsh dominfo compute1 | grep CPU
CPU(s): 4
CPU time: 32.8s
```

Connect to the VM and confirm also.

```
ada:~$ ssh osbash@192.168.122.140
osbash@192.168.122.140's password: osbash
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-57-generic x86_64)

* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage
Last login: Sat May  6 05:34:27 2017 from 192.168.122.1
```

```
osbash@compute1:~$ lscpu | grep '^CPU(s)'
CPU(s): 4
```

```
osbash@compute1:~$ cat /proc/meminfo | grep MemTotal
MemTotal: 4013452 kB
```

There is now 4 vCPUs and 4 GB of memory.
11.2 Compute node – VirtualBox

On VirtualBox change the compute node memory and CPUs (Assuming the hardware can accommodate the change).

Confirm the current vCPU and memory available at the compute node.

```
ada:~$ ssh osbash@localhost -p 2232
osbash@localhost's password:
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-57-generic x86_64)

  * Documentation:  https://help.ubuntu.com
  * Management:     https://landscape.canonical.com
  * Support:        https://ubuntu.com/advantage
Last login: Mon Sept 25 14:02:36 2017 from 10.0.2.2

osbash@compute1:~$ lscpu | grep '^CPU(s)'
CPU(s):                1

osbash@compute1:~$ cat /proc/meminfo | grep MemTotal
MemTotal:        1016164 kB
```

There is 1 vCPU and 1 GB of memory.

Before working on the memory or the CPUs on the VirtualBox VM, shutdown the VM domain.

```
ada:~$ vboxmanage controlvm compute1 poweroff
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
```

Confirm that the VM compute1 is not running.

```
ada:~$ vboxmanage list runningvms
"controller" {27b5dfa2-bff6-4fa7-a706-659175f044ee}
```

Change memory and number of vCPUs available to the VM.

```
ada:~$ vboxmanage modifyvm "compute1" --memory 4096 --cpus 4
```

Confirm these changes.

```
ada:~$ vboxmanage showvminfo compute1 | grep 'Memory size'
Memory size:     4096MB

ada:~$ vboxmanage showvminfo compute1 | grep 'Number of CPUs'
Number of CPUs:  4
```

Restart the VM compute1.

```
ada:~$ vboxmanage startvm compute1 --type headless
Waiting for VM "compute1" to power on...
VM "compute1" has been successfully started.
```
Connect to the VM and confirm also.

```
ada:~$ ssh osbash@localhost -p 2232
osbash@localhost's password:
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-57-generic x86_64)

* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage
Last login: Mon Sept 25 14:02:36 2017 from 10.0.2.2
```

```
osbash@compute1:~$ lscpu | grep '^CPU(s)'
CPU(s): 4
```

```
osbash@compute1:~$ cat /proc/meminfo | grep MemTotal
MemTotal: 4046280 kB
```

There is now 4 vCPUs and 4 GB of memory.

### 11.3 Glance - Identity Service

Download another operating system image to the controller node, say Ubuntu and add to Glance, the Image service. Ubuntu OpenStack images are stored at: [http://cloud-images.ubuntu.com](http://cloud-images.ubuntu.com). Download the OpenStack QEMU Copy On Write (QCOW2) image. The QCOW2 image format is one of the disk image formats supported by the QEMU processor emulator. It is a representation of a fixed size block device in a file. Benefits it offers over using raw dump representation include:

- Smaller file size, even on filesystems which don't support sparse files
- Copy-on-write support, where the image only represents changes made to an underlying disk image
- Snapshot support, where the image can contain multiple snapshots of the images history
- Optional zlib based compression
- Optional Advanced Encryption Standard (AES) encryption.

**Note:** While the additional image is downloaded, it may not be possible to run it later depending upon the system specifications of the training lab built, in terms of processing, memory etc..

```
osbash@controller:~$ cd img
```
root@controller:~/img$ wget http://cloud-images.ubuntu.com/xenial/current/xenial-server-cloudimg-amd64-disk1.img
Connecting to cloud-images.ubuntu.com (cloud-images.ubuntu.com)| 91.189.88.141:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 319684608 (305M) [application/octet-stream]
Saving to: 'xenial-server-cloudimg-amd64-disk1.img'
xenial-server-cloudimg-amd64-disk1 100%
[======================================>] 304.88M 668KB/s in 8m 42s

2016-12-22 20:50:04 (598 KB/s) - 'xenial-server-cloudimg-amd64-disk1.img' saved [319684608/319684608]

root@controller:~/img$ ls -la
total 325184
-rw-rw-r--  1 osbash osbash  13287936 May  7  2015 cirros-0.3.4-x86_64-disk.img
-rw-rw-r--  1 osbash osbash        63 Dec 20 10:07 cirros-0.3.4-x86_64-disk.img.md5sum
-rw-rw-r--  1 osbash osbash 319684608 Dec 21 12:12 xenial-server-cloudimg-amd64-disk1.img

Create the image as the demo User. Exit from root and set the demo variables via the demo-openrc script.

osbash@controller:~/img$ . demo-openrc.sh

osbash@controller:~/img$ openstack image create --disk-format qcow2 --container-format bare --property architecture=x86_64 --file xenial-server-cloudimg-amd64-disk1.img Ubuntu

+-----------------+------------------------------------------------------+
| Field           | Value                                                |
| checksum        | aae5d19b4e9744e3d4d63d6db5a6eae                      |
| container_format| bare                                                 |
| created_at      | 2016-12-22T20:54:05Z                                 |
| disk_format     | qcow2                                                |
| file            | /v2/images/c4ef4b37-16f5-47a2-8815-1466f103ac6/file |
| id              | c4ef4b37-16f5-47a2-8815-1466f103ac6                   |
| min_disk        | 0                                                    |
| min_ram         | 0                                                    |
| name            | Ubuntu                                               |
| owner           | 786663e8398e418ea1d08f8b6a4c91c48                     |
| properties      | architecture='x86_64'                                |
| protected       | False                                                |
| schema          | /v2/schemas/image                                    |
| size            | 319684608                                            |
| status          | active                                               |
| tags            |                                                     |
| updated_at      | 2016-12-22T20:54:06Z                                 |
| virtual_size    | None                                                 |
| visibility      | private                                              |
+-----------------+------------------------------------------------------+

osbash@controller:~/img$ cd ~
114

osbash@controller:~$ openstack image list
+--------------------------------------+--------+--------+
| ID                                   | Name   | Status |
+--------------------------------------+--------+--------+
| c4ef4b37-16f5-47a2-8815-146dfa103ac6 | Ubuntu | active |
| 6ec356b9-b15f-4592-af46-b4fb09977d16 | cirros | active |
+--------------------------------------+--------+--------+

11.4 Flavour

Create a special flavour with enlarged memory and a disk size of 3 GB (Ubuntu image is approximately 2.3 GB).

osbash@controller:~$ openstack flavor create --id 2 --vcpus 1 --ram 2048 --disk 3 m1.medium
+----------------------------+-----------+
| Field                      | Value     |
+----------------------------+-----------+
| OS-FLV-DISABLED:disabled   | False     |
| OS-FLV-EXT-DATA:ephemeral  | 0         |
| disk                       | 3         |
| id                         | 2         |
| name                       | m1.medium |
| os-flavor-access:is_public | True      |
| properties                 |           |
| ram                        | 2048      |
| rxtx_factor                | 1.0       |
| swap                       |           |
| vcpus                      | 1         |
+----------------------------+-----------+

Build new Ubuntu instance.

osbash@controller:~$ openstack server create --flavor m1.medium --image Ubuntu --nic net-id=148b32a0-ebe1-4467-ad28-62da39862e2e --security-group default --key-name mykey Ubuntu-test
+--------------------------------------+-----------------------------------------------+
| Field                                | Value                                         |
+--------------------------------------+-----------------------------------------------+
| OS-DCF:diskConfig                    | MANUAL                                        |
| OS-EXT-AZ:availability_zone          |                                               |
| OS-EXT-SRV-ATTR:host                 | None                                          |
| OS-EXT-SRV-ATTR: hypervisor_hostname | None                                          |
| OS-EXT-SRV-ATTR:instance_name        |                                               |
| OS-EXT-STS:power_state               | NOSTATE                                       |
| OS-EXT-STS:task_state                | scheduling                                    |
| OS-EXT-STS:vm_state                  | building                                      |
| OS-SRV-USG:launched_at               | None                                          |
| OS-SRV-USG:terminated_at             | None                                          |
| accessIPv4                           |                                               |
| accessIPv6                           |                                               |
| addresses                            |                                               |
| adminPass                            | jRTpXedmt9C                                  |
| config_drive                         |                                               |
| created                              | 2016-12-23T08:54:34Z                          |
| flavor                               | m1.medium (2)                                 |
| hostId                               |                                               |
| id                                   | d3943e75-12bd-430e-9199-f7fcd794be66           |
| image                                | Ubuntu (c4ef4b37-16f5-47a2-8815-146dfa103ac6) |
| key_name                             | mykey                                         |
| name                                 | Ubuntu-test                                   |
| os-extended-volumes:volumes_attached | []                                            |
| progress                             | 0                                              |
| project_id                           | 040ff9bf6990430d83c71a5765526067               |
| properties                           | [{u'name': u'default'}]                       |
| security_groups                      | [{u'name': u'default'}]                       |
| status                               | BUILD                                         |
| updated                              | 2016-12-23T08:54:34Z                          |
| user_id                              | cfbcd736f86949dbb768dd97b6797486               |
Check the status of the new VM instance.

```
osbash@controller:~$ openstack server list
```

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Status</th>
<th>Networks</th>
<th>Image Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>d3943e75-12bd-430e-9199-f7fdc794be66</td>
<td>Ubuntu-test</td>
<td>ACTIVE</td>
<td>provider=203.0.113.111</td>
<td>Ubuntu</td>
</tr>
</tbody>
</table>

Get the Console URL to access.

```
osbash@controller:~$ openstack console url show Ubuntu-test
```

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>novnc</td>
</tr>
<tr>
<td>url</td>
<td><a href="http://10.0.0.11:6080/vnc_auto.html?token=26ab4e72-7aa0-4487-81ee-a58143a3c5fa">http://10.0.0.11:6080/vnc_auto.html?token=26ab4e72-7aa0-4487-81ee-a58143a3c5fa</a></td>
</tr>
</tbody>
</table>

```
http://10.0.0.11:6080/vnc_auto.html?token=26ab4e72-7aa0-4487-81ee-a58143a3c5fa
```

Illustration 10: Ubuntu instance
This page is intentionally blank
12. Adding an additional Compute node on KVM/QEMU

The OpenStack environment can be scaled out by adding further compute nodes. In this part of the laboratory an additional node compute2 will be added to OpenStack.

Make sure there is adequate memory and drive space available to double the compute node requirements. In this case the lab was built with original values in the config.controller and config.compute1 files. If in building the lab they were adjusted to maximise the available RAM then it will not be possible to do this part of the lab without rebuilding the cluster.

12.1 Clone the Compute VM

Clone compute1 as compute2. Suspend compute1 while cloning, then execute the clone command.

```
ada:~$ virsh
Welcome to virsh, the virtualization interactive terminal.
Type: 'help' for help with commands
'quit' to quit

virsh # suspend compute1
Domain compute1 suspended

ada:~$ virt-clone --original 'compute1' --name 'compute2' --auto-clone
WARNING Setting the graphics device port to autoport, in order to avoid conflicting.
Allocating 'compute2' | 9.8 GB 00:00:32
Allocating 'compute1-sdb-clone' | 1.0 GB 00:00:03
Clone 'compute2' created successfully.
```

Review the current VMs.

```
virsh # list --all
Id  Name                           State
-----------------------------     -------------
65  controller                    running
67  compute1                      paused
-   compute2                      shut off
```

12.1.1 Start clone

Start compute2.

```
virsh # start compute2
Domain compute2 started
```

Remember this clone is still identical to compute1.
12.1.2 Connect to clone

```
ada:~$ ssh osbash@192.168.122.139
osbash@192.168.122.139's password: osbash
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-57-generic x86_64)
```

* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage

Last login: Mon Sept 25 20:12:23 2017 from 192.168.122.1
osbash@compute1:~$

12.1.3 Reconfigure clone

Edit the `/etc/network/interfaces` to reflect the IP address of `compute2` node as `10.0.0.32`.

`/etc/hosts`

```
osbash@compute1:~$ sudo vi /etc/hosts
127.0.0.1 localhost
127.0.1.1 compute2

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

# http://docs.openstack.org/mitaka/install-guide-ubuntu/environment-networking-controller.html

# controller
10.0.0.11 controller

# compute1
10.0.0.31 compute1

# compute2
10.0.0.32 compute2

# block1
10.0.0.41 block1

# object1
10.0.0.51 object1

# object2
10.0.0.52 object2
```

`/etc/hostname`

Edit the `/etc/hostname` file.

```
osbash@compute1:~$ sudo vi /etc/hostname
compute2
```
Edit the `/etc/network/interfaces` to reflect the IP address or `compute2` node as 10.0.0.32.

```bash
osbash@compute1:~$ sudo vi /etc/network/interfaces
# The loopback network interface
auto lo
iface lo inet loopback

# VirtualBox NAT -- for Internet access to VM
auto ens3
iface ens3 inet dhcp

auto ens4
iface ens4 inet static
  address 10.0.0.32
  netmask 255.255.255.0

auto ens5
iface ens5 inet manual
up ip link set dev $IFACE up
down ip link set dev $IFACE down
```

**12.1.4 Reboot instance and login to see changes**

```bash
osbash@compute1:~$ sudo shutdown --reboot now

ada:~$ ssh osbash@192.168.122.139
osbash@192.168.122.139's password: osbash
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-57-generic x86_64)

* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage
Last login: Mon Sept 25 20:33:52 2017 from 192.168.122.1
osbash@compute2:~$
```

**12.2 Start the controller and compute1**

Resume the suspended VM instance `compute1`

```bash
virsh # resume compute1
Domain compute1 resumed

virsh # list
<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>65</td>
<td>controller</td>
<td>running</td>
</tr>
<tr>
<td>67</td>
<td>compute1</td>
<td>running</td>
</tr>
<tr>
<td>68</td>
<td>compute2</td>
<td>running</td>
</tr>
</tbody>
</table>
```
It is getting a little monotonous matching up the MAC addresses to the IP addresses so here is a short program to extract the addresses and match them up.

```
ada:~$ cat <<'EOM' > ~/get_ip.sh
#!/bin/bash

CONTROLLER_MAC=`virsh --connect qemu:///system domiflist controller | grep virbr0 | awk '{print $5}'`
COMPUTE1_MAC=`virsh --connect qemu:///system domiflist compute1 | grep virbr0 | awk '{print $5}'`
COMPUTE2_MAC=`virsh --connect qemu:///system domiflist compute2 | grep virbr0 | awk '{print $5}'`

CONTROLLER_IP=`arp -e | grep $CONTROLLER_MAC | awk '{print $1}'`
COMPUTE1_IP=`arp -e | grep $COMPUTE1_MAC | awk '{print $1}'`
COMPUTE2_IP=`arp -e | grep $COMPUTE2_MAC | awk '{print $1}'`

echo "Controller IP: $CONTROLLER_IP"
echo "Compute1 IP: $COMPUTE1_IP"
echo "Compute2 IP: $COMPUTE2_IP"

EOM
```

```
ada:~$ chmod +x ~/get_ip.sh
ada:~$ ~/get_ip.sh
Controller IP: 192.168.122.82
Compute1 IP: 192.168.122.140
Compute2 IP: 192.168.122.139
```

### 12.2.1 Adjust host table in the compute1

Add compute2 to the /etc/hosts file of compute1.

```
ada:~$ ssh osbash@192.168.122.140
The authenticity of host '192.168.122.140 (192.168.122.140)' can't be established.
ECDSA key fingerprint is
SHA256:pD7/Z+rGK7Rs5YiHEQt80j4UDc5SnYkTs+Ahd+pD33M.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.122.140' (ECDSA) to the list of known hosts.
```

```
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-57-generic x86_64)

* Documentation:  https://help.ubuntu.com
* Management:     https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage
Last login: Mon Sept 25 20:12:23 2017 from 192.168.122.1
osbash@compute1:~$```
OpenStack Training Laboratory

osbash@compute1:~$ sudo vi /etc/hosts
127.0.0.1       localhost
127.0.1.1       compute1-lo

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

# http://docs.openstack.org/mitaka/install-guide-ubuntu/environment-networking-controller.html

# controller
10.0.0.11       controller

# compute1
10.0.0.31       compute1

# compute2
10.0.0.32       compute2

# block1
10.0.0.41       block1

# object1
10.0.0.51       object1

# object2
10.0.0.52       object2

12.2.2 Adjust the host table of the controller

Add compute2 to the /etc/hosts file.

ada:~$ ssh osbash@192.168.122.82
osbash@192.168.122.82’s password: osbash
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-57-generic x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/advantage
Last login: Mon Sept 25 08:57:48 2017 from 192.168.122.1
osbash@controller:~$
osbash@controller:~$ sudo vi /etc/hosts
127.0.0.1       localhost
127.0.1.1       controller-lo

# The following lines are desirable for IPv6 capable hosts
::1       localhost ip6-localhost ip6-loopback
ff02::1     ip6-allnodes
ff02::2     ip6-allrouters

# http://docs.openstack.org/mitaka/install-guide-ubuntu/environment-networking-controller.html

# controller
10.0.0.11       controller

# compute1
10.0.0.31       compute1

# compute2
10.0.0.32       compute2

# block1
10.0.0.41       block1

# object1
10.0.0.51       object1

# object2
10.0.0.52       object2

12.2.3 Configure compute2

Change the local IP address in /etc/nova/nova.conf.

osbash@compute2:~$ sudo sed -i.bak 's/10.0.0.31/10.0.0.32/'
/etc/nova/nova.conf

osbash@compute2:~$ sudo diff /etc/nova/nova.conf.bak /etc/nova/nova.conf
4c4
< my_ip = 10.0.0.31
---
> my_ip = 10.0.0.32

Restart the Compute service on compute2.

osbash@compute2:~$ sudo systemctl restart nova-compute

osbash@compute2:~$ sudo systemctl status nova-compute | head -3
* nova-compute.service - OpenStack Compute
    Loaded: loaded (/lib/systemd/system/nova-compute.service; enabled; vendor preset: enabled)
    Active: active (running) since Thu 2017-01-05 14:12:27 UTC; 40s ago

12.3 Check the controller for compute2

Check OpenStack Compute service to see if compute2 has registered.

```
osbash@controller:~$ . admin-openrc.sh

osbash@controller:~$ openstack compute service list
```

```
+----+------------------+------------+----------+---------+-------+----------------------------+
| ID | Binary           | Host       | Zone     | Status  | State | Updated At                 |
+----+------------------+------------+----------+---------+-------+----------------------------+
| 3  | nova-consoleauth | controller | internal | enabled | up    | 2017-01-05T21:15:07.000000  |
| 4  | nova-scheduler   | controller | internal | enabled | up    | 2017-01-05T21:15:00.000000  |
| 5  | nova-conductor   | controller | internal | enabled | up    | 2017-01-05T21:15:00.000000  |
| 6  | nova-compute     | compute1   | nova     | enabled | up    | 2017-01-05T21:15:08.000000  |
| 7  | nova-compute     | compute2   | nova     | enabled | up    | 2017-01-05T21:15:06.000000  |
+----+------------------+------------+----------+---------+-------+----------------------------+
```
This page is intentionally blank
13. Adding an additional Compute node on VirtualBox

The OpenStack environment can be scaled out by adding further compute nodes. In this part of the laboratory an additional node compute2 will be added to OpenStack.

Make sure there is adequate memory and drive space available to double the compute node requirements. In this case the lab was built with original values in the config.controller and config.compute1 files. If in building the lab they were adjusted to maximise the available RAM then it will not be possible to do this part of the lab without rebuilding the cluster.

13.1 Clone the Compute VM

Clone compute1 as compute2.

```
ada:~$ vboxmanage clonevm "compute1" --name "compute2" --register
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
Machine has been successfully cloned as "compute2"
```

Review the current VMs.

```
ada:~$ vboxmanage list vms
"controller" {27b5dfa2-bff6-4fa7-a706-659175f044ee}
"compute1" {1a5a56e5-f5dc-4fc6-b588-79256da09962}
"compute2" {42f888bf-9022-4202-b042-ac0a1f5db2b}
```

Modify the access port.

```
ada:~$ vboxmanage modifyvm "compute2" --natpfi delete ssh
ada:~$ vboxmanage modifyvm "compute2" --natpfi ssh,tcp,127.0.0.1,2233,,22
```

13.1.1 Start clone

Start compute2.

```
ada:~$ vboxmanage startvm "compute2" --type headless
Waiting for VM "compute2" to power on...
VM "compute2" has been successfully started.
```

Remember this clone is now on port 2233.

13.1.2 Connect to clone

```
ada:~$ ssh osbash@localhost -p2233
osbash@localhost's password:
osbash@compute1:~$
```

02 Oct 2017 OpenStack Pike
13.1.3 Reconfigure clone

In the /etc/hosts file edit the 127.0.1.1 entry and add compute2 with IP address 10.0.0.32.

/etc/hosts

    osbash@compute1:~$ sudo vi /etc/hosts
    127.0.0.1 localhost
    127.0.1.1 compute2-1o
    # The following lines are desirable for IPv6 capable hosts
    ::1     localhost ip6-localhost ip6-loopback
    ff02::1 ip6-allnodes
    ff02::2 ip6-allrouters
    #------------------------------------------------------------------------
    # http://docs.openstack.org/mitaka/install-guide-ubuntu/environment-networking-controller.html
    #------------------------------------------------------------------------
    # controller
    10.0.0.11 controller
    # compute1
    10.0.0.31 compute1
    # compute2
    10.0.0.32 compute2
    # block1
    10.0.0.41 block1
    # object1
    10.0.0.51 object1
    # object2
    10.0.0.52 object2

/etc/hostname

    Edit the /etc/hostname file.
    osbash@compute1:~$ sudo vi /etc/hostname
    compute2
OpenStack Training Laboratory

/etc/network/interfaces

Edit the /etc/network/interfaces to reflect the IP address or compute2 node as 10.0.0.32.

```
osbash@compute1:~$ sudo vi /etc/network/interfaces
# The loopback network interface
auto lo
iface lo inet loopback

# VirtualBox NAT -- for Internet access to VM
auto enp0s3
iface enp0s3 inet dhcp

auto enp0s8
iface enp0s8 inet static
  address 10.0.0.32
  netmask 255.255.255.0

auto enp0s9
iface enp0s9 inet manual
up ip link set dev $IFACE up
down ip link set dev $IFACE down```

13.1.4 Reboot instance and login to see changes

```
osbash@compute1:~$ sudo shutdown --reboot now

ada:~$  ssh osbash@localhost -p2233
osbash@localhost's password: osbash
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-57-generic x86_64)

* Documentation:  https://help.ubuntu.com
* Management:     https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage
Last login: Mon Sept 25 13:53:33 2017 from 10.0.2.2

osbash@compute2:~$
```

13.2 Start the controller and compute1

```
ada:~$  vboxmanage startvm "controller" --type headless
Waiting for VM "controller" to power on...
VM "controller" has been successfully started.

ada:~$  vboxmanage startvm "compute1" --type headless
Waiting for VM "compute1" to power on...
VM "compute1" has been successfully started.

ada:~$  vboxmanage list runningvms
"controller" {27b5dfa2-bff6-4fa7-a706-659175f044ee}
"compute1" {1a5a56e5-f5dc-4fc6-b588-79256da09962}
"compute2" {42f888bf-9022-4202-b042-acc0af5db2b}
```
13.2.1 Adjust host table in the compute1

Add compute2 to the /etc/hosts file.

```bash
osbash@compute1:~$ sudo vi /etc/hosts
```

```
127.0.0.1   localhost
127.0.1.1   compute1-lo

# The following lines are desirable for IPv6 capable hosts
::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
#--------------------------------
# http://docs.openstack.org/mitaka/install-guide-ubuntu/environment-networking-controller.html
#--------------------------------
# controller
10.0.0.11   controller

# compute1
10.0.0.31   compute1

# compute2
10.0.0.32   compute2

# block1
10.0.0.41   block1

# object1
10.0.0.51   object1

# object2
10.0.0.52   object2
```

13.2.2 Adjust the host table of the controller

Add compute2 to the /etc/hosts file.

```bash
osbash@controller:~$ sudo vi /etc/hosts
```

```
127.0.0.1   localhost
127.0.1.1   controller-lo

# The following lines are desirable for IPv6 capable hosts
::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
#--------------------------------
# http://docs.openstack.org/mitaka/install-guide-ubuntu/environment-networking-controller.html
#--------------------------------
# controller
10.0.0.11   controller

# compute1
10.0.0.31   compute1

# compute2
10.0.0.32   compute2

# block1
10.0.0.41   block1

# object1
10.0.0.51   object1

# object2
10.0.0.52   object2
```
13.2.3 Configure compute2

Change the local IP address in `/etc/nova/nova.conf`.

```
osbash@compute2:~$ sudo sed -i.bak 's/10.0.0.31/10.0.0.32/' /etc/nova/nova.conf
```

```
osbash@compute2:~$ sudo diff /etc/nova/nova.conf.bak /etc/nova/nova.conf
4c4
< my_ip = 10.0.0.31
---
> my_ip = 10.0.0.32
```

Restart the Compute service on `compute2`.

```
osbash@compute2:~$ sudo systemctl restart nova-compute
```

```
osbash@compute2:~$ sudo systemctl status nova-compute | head -3
* nova-compute.service - OpenStack Compute
  Loaded: loaded (/lib/systemd/system/nova-compute.service; enabled; vendor preset: enabled)
  Active: active (running) since Thu 2017-01-05 14:12:27 UTC; 40s ago
```

13.3 Check the controller for compute2

Check OpenStack Compute service to see if `compute2` has registered.

```
osbash@controller:~$ . admin-openrc.sh
```

```
osbash@controller:~$ openstack compute service list
+----+------------------+------------+----------+---------+-------+----------------------------+
<table>
<thead>
<tr>
<th>ID</th>
<th>Binary</th>
<th>Host</th>
<th>Zone</th>
<th>Status</th>
<th>State</th>
<th>Updated At</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>nova-consoleauth</td>
<td>controller</td>
<td>internal</td>
<td>enabled</td>
<td>up</td>
<td>2017-01-05T14:17:24.000000</td>
</tr>
<tr>
<td>4</td>
<td>nova-conductor</td>
<td>controller</td>
<td>internal</td>
<td>enabled</td>
<td>up</td>
<td>2017-01-05T14:17:23.000000</td>
</tr>
<tr>
<td>5</td>
<td>nova-scheduler</td>
<td>controller</td>
<td>internal</td>
<td>enabled</td>
<td>up</td>
<td>2017-01-05T14:17:24.000000</td>
</tr>
<tr>
<td>6</td>
<td>nova-compute</td>
<td>compute1</td>
<td>nova</td>
<td>enabled</td>
<td>up</td>
<td>2017-01-05T14:17:23.000000</td>
</tr>
<tr>
<td>7</td>
<td>nova-compute</td>
<td>compute2</td>
<td>nova</td>
<td>enabled</td>
<td>up</td>
<td>2017-01-05T14:17:19.000000</td>
</tr>
</tbody>
</table>
+----+------------------+------------+----------+---------+-------+----------------------------+
```

02 Oct 2017 OpenStack Pike netLabs/UG
This page is intentionally blank
14. **KVM/QEMU restarting procedure after shutdown**

To restart the testbed after the host shutdown requires that the networks are enabled first.

```
ada:~$ virsh
Welcome to virsh, the virtualization interactive terminal.
Type: 'help' for help with commands
'quit' to quit
```

14.1 **List and enable networks**

```
virsh # net-list --all
Name                State      Autostart     Persistent
----------------------------------------------------------
default              active     yes           yes
labs-mgmt            inactive   no            yes
labs-provider        inactive   no            yes
```

Enable the two inactive networks.

```nvirsh # net-start labs-mgmt
Network labs-mgmt started
```

```nvirsh # net-start labs-provider
Network labs-provider started
```

Set the networks to auto start in future after restart.

```nvirsh # net-autostart labs-mgmt
Network labs-mgmt marked as autostarted
```

```nvirsh # net-autostart labs-provider
Network labs-provider marked as autostarted
```

Note that networks are now active and set to auto start after future reboots.

```
virsh # net-list --all
Name                State      Autostart     Persistent
----------------------------------------------------------
default              active     yes           yes
labs-mgmt            active     yes           yes
labs-provider        active     yes           yes
```
14.2 Start the nodes

Start each of the nodes.

```
virsh # list --all
Id    Name                           State
----------------------------------------------------
-     compute1                       shut off
-     controller                     shut off

virsh # start controller
Domain controller started

virsh # start compute1
Domain compute1 started
```

Confirm nodes are running.

```
virsh # list
Id    Name                           State
----------------------------------------------------
 1     controller                     running
 2     compute1                       running
```
15. Working with the Horizon dashboard

15.1 Accessing Horizon on KVM/QEMU testbed

To access Horizon Dashboard on the KVM/QEMU testbed, it is necessary to browse from the KVM/QEMU host. As this is a headless server it is therefore necessary to redirect the browser to the workstation via SSH X11 forwarding.

```
ada:~$ ssh -MY alovelace@virtserver
Welcome to Ubuntu 16.10 (GNU/Linux 4.8.0-32-generic x86_64)
  * Documentation: https://help.ubuntu.com
  * Management: https://landscape.canonical.com
  * Support: https://ubuntu.com/advantage

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

Last login: Sat May 6 07:30:50 2017 from 192.168.89.2
```

```
ada:~$ sudo apt-get install firefox
```

```
ada:~$ firefox http://10.0.0.11/horizon
```

*Illustration 11: Horizon login - KVM/QEMU testbed*
15.2 Accessing Horizon on the VirtualBox testbed

Browse to:

http://localhost:8888/horizon

*Illustration 12: Horizon login - VirtualBox testbed*
15.3 Logging in

Two accounts are configured: admin with the password `admin_user_secret` and demo with the password `demo_user_pass`. The default domain required for login is default. These and other passwords are configured in `config/credentials`.

<table>
<thead>
<tr>
<th>Project name</th>
<th>Username</th>
<th>Password</th>
<th>User role name</th>
</tr>
</thead>
<tbody>
<tr>
<td>admin</td>
<td>admin</td>
<td>admin_user_secret</td>
<td>admin</td>
</tr>
<tr>
<td>demo</td>
<td>demo</td>
<td>demo_user_pass</td>
<td>user</td>
</tr>
</tbody>
</table>

*Illustration 13: Admin opening dashboard screen*
15.4 ‘admin’ User functions

15.4.1 Create a new project

Click **Create Project**

- Name: **The Difference Engine**
- Description: **A new project via horizon.**

Click **Create Project**.

*Illustration 14: Create Project*
15.4.2 Create a flavour

Click Admin ▶ Compute ▶ Flavours, then +Create Flavour.

- Name: m1.small
- ID: Auto
- VCPUs: 1
- RAM (MB): 2048
- Root Disk (GB): 2

Click Create Flavour.

Illustration 15: Create Flavour
15.4.3 Create a new user

Click **Identity** ➤ **Users**, then +**Create** User.

- Username: **alovelace**  Description: A new Ada Lovelace user via horizon.
- Email: **ada@lovelace.com**
- Password: **babbage**  Confirm Password: **babbage**
- Primary Project: **The Difference Engine**
- Role: **user**

Click **Create User**.

*Illustration 16: Create User*
15.5  **Project user functions**

Users within the Project can create instances, networks, etc. within the overall scope established by the admin. In fact the admin cannot see the instances established by the Project user.

Log out and log back in as user *alovelace*.

Click *admin* in top right corner and then *Sign Out*.

Login as Username: *alovelace* Password: *babbage*.

![Illustration 17: Project User opening dashboard screen](image-url)
15.5.1 Create a Security Group

Click *Project ► Network ► Security Groups* then *Create Security Group*.

Name:  *The Difference Engine SG*  
Description:  *The Difference Engine Security group to allow SSH and ICMP.*

Click *Create Security Group*.

Once created check the tick box for *The Difference Engine SG* and click *Manage Rules*. Click *Add Rule*.

Add

Rule:  *All ICMP*  
Direction:  *Ingress*  
Remote:  *CIDR*  
CIDR:  *0.0.0.0/0*

Rule:  *SSH*  
Remote:  *CIDR*  
CIDR:  *0.0.0.0/0*
### Illustration 19: Adding rules

<table>
<thead>
<tr>
<th>Direction</th>
<th>Ether Type</th>
<th>IP Protocol</th>
<th>Port Range</th>
<th>Remote IP Prefix</th>
<th>Remote Security Group</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ingress</td>
<td>IPv6</td>
<td>Any</td>
<td>Any</td>
<td>-</td>
<td>default</td>
<td></td>
</tr>
<tr>
<td>Egress</td>
<td>IPv6</td>
<td>Any</td>
<td>Any</td>
<td>::/0</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Ingress</td>
<td>IPv4</td>
<td>Any</td>
<td>Any</td>
<td>-</td>
<td>default</td>
<td></td>
</tr>
<tr>
<td>Egress</td>
<td>IPv4</td>
<td>Any</td>
<td>Any</td>
<td>0.0.0.0/0</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Ingress</td>
<td>IPv4</td>
<td>ICMP</td>
<td>Any</td>
<td>0.0.0.0/0</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Ingress</td>
<td>IPv4</td>
<td>TCP</td>
<td>22 (SSH)</td>
<td>0.0.0.0/0</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>
15.5.2 Create an instance

Click **Compute ▶ Instances**, then **Launch Instance**.

Instance name: **Engine1**

Availability Zone: **Nova**

Count: **1**

Click **Next**.

Click the + symbol beside **cirros** in the **Available** section to move it to **Allocated**.
Illustration 21: Instance Launch - Source

Click Next.
Click the + symbol beside **m1.small** flavour in the *Available* section to move it to *Allocated*.

![Illustration 22: Launch Instance - Flavour](image)

Click the + symbol beside **The Difference Engine SG** in the *Available* section to move it to *Allocated* and click the – symbol beside **default** to move it from *Allocated* to *Available*.

![Illustration 23: Add the Security Group](image)
Click **Launch Instance**.

![Instance Launched](image)

### Illustration 24: Instance Launched

#### 15.6 Run the nat_tables.sh script

Establish a NAT for the provider network to access the Internet. Before running the script set the network, i.e. `virbr2` for KVM/QEMU or `vboxnet1` for VirtualBox. Also set the local Internet interface on the hypervisor host.

```bash
ada:~$ sudo $OS_LAB/nat_tables.sh
[sudo] password for alovelace: babbage

echo "1" > /proc/sys/net/ipv4/ip_forward

Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source     destination
  0     0 MASQUERADE  all  --  any    enp0s3  anywhere  anywhere

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source     destination
  0     0 ACCEPT     all  --  enp0s3 virbr2  anywhere   anywhere     state RELATED,ESTABLISHED
  0     0 ACCEPT     all  --  virbr2 enp3s0  anywhere   anywhere
```

#### 15.7 Connect to the Instance

Get the IP address of the instance from the instance dashboard. Confirm connection to the `engine1` VM.

```bash
ada:~$ ssh cirros@203.0.113.103
The authenticity of host '203.0.113.103 (203.0.113.108)' can't be established.
RSA key fingerprint is
SHA256:Y/YgWK7vcObPGVndX+taxKUfw/s17uU1LT1T6GNUfk.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '203.0.113.108' (RSA) to the list of known hosts.
cirros@203.0.113.108's password: cubswin:)
$

$ hostname
eengine1
```
16. Creating networks

Now that creating instances is mastered consider the creation of networks. The diagram in Illustration 25 demonstrates a simple network with 4 hosts, two on the default provider network and two on a new private network which is connected to the provider network via a router. Here is an explanation of the process for the creation of the additional private network, hosts, a router and connecting the networks.

Steps to be followed are:

1. Enable the admin-openrc variables
2. Create a flavour
3. Enable the demo-openrc variables
4. Add port 22 (SSH) and ICMP to default security group
5. Create private network
6. Extract provider and private network UUIDs
7. Create hosts on the provider network
8. Create hosts on the private network
9. Create a router
10. Add subnet to the router
11. Add a default route to the router via 203.0.113.1
12. Add a route on the host to the private network.
16.1 Initial configuration

Before getting into the networking element enable the \textit{admin-openrc} variables and create a flavour as demonstrated already on page 83.

Now enable the \textit{demo-openrc} variables and add port 22 (SSH) and ICMP to \textit{default} security group as demonstrated on page 89.

16.2 Create private network

Create a private network and assign network information to it.

\begin{verbatim}
osbash@controller:~$ openstack network create PRIV-NET

osbash@controller:~$ openstack subnet create --network PRIV-NET \
    --subnet-range 192.168.95.0/24 --gateway 192.168.95.1 --dhcp \
    --allocation-pool start=192.168.95.10,end=192.168.95.20 \
    --dns-nameserver 8.8.8.8 PRIV-SUBNET
\end{verbatim}

Extract provider and private network UUIDs

\begin{verbatim}
osbash@controller:~$ openstack network list | grep provider | awk '{print $2}'
1ad8799b-8d9a-4ddd-801f-942da3549ee4

osbash@controller:~$ openstack network list | grep PRIV-NET | awk '{print $2}'
34477f0c-cedc-4a0c-a7ab-66439a5c709b
\end{verbatim}

16.3 Create hosts on the provider and private networks

Launch four instances, two on the provider network and two on the private network as demonstrated on page 94.

16.4 Create a router

Create a router.

\begin{verbatim}
osbash@controller:~$ openstack router create router1

osbash@controller:~$ openstack router set --external-gateway=provider router1

osbash@controller:~$ openstack router add subnet router1 PRIV-SUBNET
\end{verbatim}

Add a default route to the router via 203.0.113.1

\begin{verbatim}
osbash@controller:~$ openstack router set router1 --route \
    destination=0.0.0.0/0,gateway=203.0.113.1
\end{verbatim}
16.5 **Add a route on the hypervisor to the private network**

Add a static route from the hypervisor host to the private network. First determine the IP address assigned to the OpenStack router provider interface.

```
osbash@controller:~$ openstack router show router1 | grep external_gateway_info | awk -F "" '{print $16}'
```

203.0.113.104

Then add the static route to the private network.

```
ada:~$ sudo ip route add 192.168.95.0/24 metric 1 nexthop via 203.0.113.104
```

16.6 **Test the configuration**

From the hypervisor host ping the four hosts.

```
ada:~$ ping -c1 203.0.113.102
PING 203.0.113.102 (203.0.113.102) 56(84) bytes of data.
64 bytes from 203.0.113.102: icmp_seq=1 ttl=64 time=0.960 ms
--- 203.0.113.102 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.960/0.960/0.960/0.000 ms
```

```
ada:~$ ping -c1 203.0.113.103
PING 203.0.113.103 (203.0.113.103) 56(84) bytes of data.
64 bytes from 203.0.113.103: icmp_seq=1 ttl=64 time=2.10 ms
--- 203.0.113.103 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 2.101/2.101/2.101/0.000 ms
```

```
ada:~$ ping -c1 192.168.95.12
PING 192.168.95.12 (192.168.95.12) 56(84) bytes of data.
64 bytes from 192.168.95.12: icmp_seq=1 ttl=63 time=0.866 ms
--- 192.168.95.12 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.866/0.866/0.866/0.000 ms
```

```
ada:~$ ping -c1 195.168.95.17
PING 195.168.95.17 (195.168.95.17) 56(84) bytes of data.
64 bytes from 195.168.95.17: icmp_seq=1 ttl=235 time=238 ms
--- 195.168.95.17 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 238.718/238.718/238.718/0.000 ms
```

Connect to one of the hosts on the private network and confirm connectivity to the Internet.

```
ada:~$ ssh cirros@192.168.95.12
cirros@192.168.95.12's password: cubswin:
```

```
$ ping -c1 8.8.8.8
PING 8.8.8.8 (8.8.8.8): 56 data bytes
64 bytes from 8.8.8.8: seq=0 ttl=54 time=234.032 ms
--- 8.8.8.8 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 234.032/234.032/234.032 ms
```
16.7 Review topology on the Horizon dashboard

Login as user demo password: `demo_user_pass` and select:

Network > Network Topology > Topology : Normal

*Illustration 26: Network topology*

Network > Network Topology > Graph : Toggle labels

*Illustration 27: Network graph*
16.8 Scripting the operation

The script in Appendix 5 - Script to launch a network with VMs can be used to completely build the network and the associated hosts.

Make sure as the demo user that there is no existing routers, networks (except for the default provider network) or hosts. Also remove the entries for SSH and ICMP from the default security group. As the admin user make sure that the flavour m1.nano is removed.

Run the script.

```
oshbash@controller:~$ ./network_launch.sh
admin-openrc script created
demo-openrc script created
```

Setting admin-openrc variables

Creating flavour m1.nano

```
+----------------------------+---------+
| Field                      | Value   |
+----------------------------+---------+
| OS-FLV-DISABLED:disabled   | False   |
| OS-FLV-EXT-DATA:ephemeral  | 0       |
| disk                       | 1       |
| id                         | 0       |
| name                       | m1.nano |
| os-flavor-access:is_public | True    |
| properties                 |         |
| ram                        | 1024    |
| rxtx_factor                | 1.0     |
| swap                       |         |
| vcpus                      | 1       |
```

Setting demo-openrc variables

Adding port 22 (SSH) and ICMP to default security group

```
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| created_at        | 2017-02-17T09:14:26Z                 |
| description       |                                      |
| direction         | ingress                              |
| ethertype         | IPv4                                 |
| headers           |                                      |
| id                | 26576af5-65d7-4e5f-9f3f-055cbbffe34e |
| port_range_max    | 22                                   |
| port_range_min    | 22                                   |
| project_id        | f5a2b881391e4170b1649c7343e0b361     |
| project_id        | f5a2b881391e4170b1649c7343e0b361     |
| protocol          | tcp                                  |
| remote_group_id   | None                                 |
| remote_ip_prefix  | 0.0.0.0/0                            |
| revision_number   | 1                                    |
| security_group_id | 7f4c8b8c-1f55-4273-bc03-60d2ba39fb42 |
| updated_at        | 2017-02-17T09:14:26Z                 |
+-------------------+--------------------------------------+
<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>created_at</td>
<td>2017-02-17T09:14:28Z</td>
</tr>
<tr>
<td>description</td>
<td></td>
</tr>
<tr>
<td>direction</td>
<td>ingress</td>
</tr>
<tr>
<td>ethertype</td>
<td>IPv4</td>
</tr>
<tr>
<td>headers</td>
<td></td>
</tr>
<tr>
<td>id</td>
<td>4dd564a0-3615-484f-b915-d22b4df8016d</td>
</tr>
<tr>
<td>port_range_max</td>
<td>None</td>
</tr>
<tr>
<td>port_range_min</td>
<td>None</td>
</tr>
<tr>
<td>project_id</td>
<td>f5a2b881391e4170b1649c7343e0b361</td>
</tr>
<tr>
<td>project_id</td>
<td></td>
</tr>
<tr>
<td>protocol</td>
<td>icmp</td>
</tr>
<tr>
<td>remote_group_id</td>
<td>None</td>
</tr>
<tr>
<td>remote_ip_prefix</td>
<td>0.0.0.0/0</td>
</tr>
<tr>
<td>revision_number</td>
<td>1</td>
</tr>
<tr>
<td>security_group_id</td>
<td>7f4c8b8c-1f55-4273-bc03-60d2ba39fb42</td>
</tr>
<tr>
<td>updated_at</td>
<td>2017-02-17T09:14:28Z</td>
</tr>
</tbody>
</table>

Creating private network PRIV-NET

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>admin_state_up</td>
<td>UP</td>
</tr>
<tr>
<td>availability_zone_hints</td>
<td></td>
</tr>
<tr>
<td>availability_zones</td>
<td></td>
</tr>
<tr>
<td>created_at</td>
<td>2017-02-17T09:14:29Z</td>
</tr>
<tr>
<td>description</td>
<td></td>
</tr>
<tr>
<td>headers</td>
<td></td>
</tr>
<tr>
<td>id</td>
<td>18a99a37-c0bd-4d8a-bdcd-9fad01c267a9</td>
</tr>
<tr>
<td>ipv4_address_scope</td>
<td>None</td>
</tr>
<tr>
<td>ipv6_address_scope</td>
<td>None</td>
</tr>
<tr>
<td>mtu</td>
<td>1450</td>
</tr>
<tr>
<td>name</td>
<td>PRIV-NET</td>
</tr>
<tr>
<td>port_security_enabled</td>
<td>True</td>
</tr>
<tr>
<td>project_id</td>
<td>f5a2b881391e4170b1649c7343e0b361</td>
</tr>
<tr>
<td>project_id</td>
<td></td>
</tr>
<tr>
<td>revision_number</td>
<td>3</td>
</tr>
<tr>
<td>router:external</td>
<td>Internal</td>
</tr>
<tr>
<td>shared</td>
<td>False</td>
</tr>
<tr>
<td>status</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>subnets</td>
<td></td>
</tr>
<tr>
<td>tags</td>
<td>[]</td>
</tr>
<tr>
<td>updated_at</td>
<td>2017-02-17T09:14:29Z</td>
</tr>
<tr>
<td>Field</td>
<td>Value</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>allocation_pools</td>
<td>192.168.95.10-192.168.95.20</td>
</tr>
<tr>
<td>cidr</td>
<td>192.168.95.0/24</td>
</tr>
<tr>
<td>created_at</td>
<td>2017-02-17T09:14:31Z</td>
</tr>
<tr>
<td>description</td>
<td></td>
</tr>
<tr>
<td>dns_nameservers</td>
<td>8.8.8.8</td>
</tr>
<tr>
<td>enable_dhcp</td>
<td>True</td>
</tr>
<tr>
<td>gateway_ip</td>
<td>192.168.95.1</td>
</tr>
<tr>
<td>headers</td>
<td></td>
</tr>
<tr>
<td>host_routes</td>
<td></td>
</tr>
<tr>
<td>id</td>
<td>de061a25-190c-4cfa-ac1c-444445e963b6</td>
</tr>
<tr>
<td>ip_version</td>
<td>4</td>
</tr>
<tr>
<td>ipv6_address_mode</td>
<td>None</td>
</tr>
<tr>
<td>ipv6_ra_mode</td>
<td>None</td>
</tr>
<tr>
<td>name</td>
<td>PRIV-SUBNET</td>
</tr>
<tr>
<td>network_id</td>
<td>18a99a37-c0bd-4d8a-bdcd-9fad01c267a9</td>
</tr>
<tr>
<td>project_id</td>
<td>f5a2b881391e4170b1649c7343e0b361</td>
</tr>
<tr>
<td>project_id</td>
<td>f5a2b881391e4170b1649c7343e0b361</td>
</tr>
<tr>
<td>revision_number</td>
<td>2</td>
</tr>
<tr>
<td>service_types</td>
<td>[]</td>
</tr>
<tr>
<td>subnetpool_id</td>
<td>None</td>
</tr>
<tr>
<td>updated_at</td>
<td>2017-02-17T09:14:31Z</td>
</tr>
</tbody>
</table>

Extracting provider and PRIV-NET network UUIDs

Provider: 1ad8799b-8d9a-43dd-801f-942da3549ee4

PRIV-NET: 18a99a37-c0bd-4d8a-bdcd-9fad01c267a9

Create hosts on provider network

Creating and launching instance host1 with:

Flavour: m1.nano
Image: cirros
Network UUID=1ad8799b-8d9a-43dd-801f-942da3549ee4
Security group: default

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS-DCF:diskConfig</td>
<td>MANUAL</td>
</tr>
<tr>
<td>OS-EXT-AZ:availability_zone</td>
<td></td>
</tr>
<tr>
<td>OS-EXT-STS:power_state</td>
<td>NOSTATE</td>
</tr>
<tr>
<td>OS-EXT-STS:task_state</td>
<td>scheduling</td>
</tr>
<tr>
<td>OS-EXT-STS:vm_state</td>
<td>building</td>
</tr>
<tr>
<td>OS-SRV-USG:launched_at</td>
<td>None</td>
</tr>
<tr>
<td>OS-SRV-USG:terminated_at</td>
<td>None</td>
</tr>
<tr>
<td>accessIPv4</td>
<td></td>
</tr>
<tr>
<td>accessIPv6</td>
<td></td>
</tr>
<tr>
<td>addresses</td>
<td></td>
</tr>
<tr>
<td>adminPass</td>
<td>vHKpBFU3szS4</td>
</tr>
<tr>
<td>config_drive</td>
<td></td>
</tr>
<tr>
<td>created</td>
<td>2017-02-17T09:14:38Z</td>
</tr>
<tr>
<td>flavor</td>
<td>m1.nano (0)</td>
</tr>
<tr>
<td>hostId</td>
<td></td>
</tr>
<tr>
<td>id</td>
<td>0905d666-353a-4f92-bfco-4f630b69564</td>
</tr>
<tr>
<td>image</td>
<td>cirros (6846e263-d0c9-46da-b643-4d934d80b36f)</td>
</tr>
<tr>
<td>key_name</td>
<td>None</td>
</tr>
<tr>
<td>name</td>
<td>host1</td>
</tr>
<tr>
<td>os-extended-volumes:volumes_attached</td>
<td>[]</td>
</tr>
<tr>
<td>progress</td>
<td>0</td>
</tr>
<tr>
<td>project_id</td>
<td>f5a2b881391e4170b1649c7343e0b361</td>
</tr>
<tr>
<td>properties</td>
<td></td>
</tr>
<tr>
<td>security_groups</td>
<td>[{u'name': u'default'}]</td>
</tr>
<tr>
<td>status</td>
<td>BUILD</td>
</tr>
<tr>
<td>updated</td>
<td>2017-02-17T09:14:39Z</td>
</tr>
<tr>
<td>user_id</td>
<td>4bc1f71e027348a6b81ab62f393bb9c9d8</td>
</tr>
</tbody>
</table>
Creating and launching instance host2 with:

Flavour: m1.nano
Image: cirros
Network UUID=1ad8799b-8d9a-4ddd-801f-942da3549ee4
Security group: default

```
<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS-DCF:diskConfig</td>
<td>MANUAL</td>
</tr>
<tr>
<td>OS-EXT-AZ:availability_zone</td>
<td></td>
</tr>
<tr>
<td>OS-EXT-STS:power_state</td>
<td>NOSTATE</td>
</tr>
<tr>
<td>OS-EXT-STS:task_state</td>
<td>scheduling</td>
</tr>
<tr>
<td>OS-EXT-STS:vm_state</td>
<td>building</td>
</tr>
<tr>
<td>OS-SRV-USG:launched_at</td>
<td>None</td>
</tr>
<tr>
<td>OS-SRV-USG:terminated_at</td>
<td>None</td>
</tr>
<tr>
<td>accessIPv4</td>
<td></td>
</tr>
<tr>
<td>accessIPv6</td>
<td></td>
</tr>
<tr>
<td>addresses</td>
<td></td>
</tr>
<tr>
<td>config_drive</td>
<td></td>
</tr>
<tr>
<td>created</td>
<td>2017-02-17T09:14:44Z</td>
</tr>
<tr>
<td>flavor</td>
<td>m1.nano (0)</td>
</tr>
<tr>
<td>hostId</td>
<td></td>
</tr>
<tr>
<td>id</td>
<td>da5f5338-296e-4053-b980-bc6718f0d1ab</td>
</tr>
<tr>
<td>image</td>
<td>cirros (6846e263-d0c9-46da-b643-4e95340dddef8)</td>
</tr>
<tr>
<td>key_name</td>
<td>None</td>
</tr>
<tr>
<td>name</td>
<td>host2</td>
</tr>
<tr>
<td>os-extended-volumes:volumes_attached</td>
<td>[]</td>
</tr>
<tr>
<td>progress</td>
<td>0</td>
</tr>
<tr>
<td>project_id</td>
<td>f5a2b881391e4170b1649c7343e0b361</td>
</tr>
<tr>
<td>properties</td>
<td></td>
</tr>
<tr>
<td>security_groups</td>
<td>[{u'name': u'default'}]</td>
</tr>
<tr>
<td>status</td>
<td>BUILD</td>
</tr>
<tr>
<td>updated</td>
<td>2017-02-17T09:14:44Z</td>
</tr>
<tr>
<td>user_id</td>
<td>4bc1f71e027348a6b81ab62f93bbc9d8</td>
</tr>
</tbody>
</table>
```

Create hosts on PRIV-NET network

Creating and launching instance host3 with:

Flavour: m1.nano
Image: cirros
Network UUID=18a99a37-c0bd-4d8a-bdc7-9fad01c267a9
Security group: default

```
<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS-DCF:diskConfig</td>
<td>MANUAL</td>
</tr>
<tr>
<td>OS-EXT-AZ:availability_zone</td>
<td></td>
</tr>
<tr>
<td>OS-EXT-STS:power_state</td>
<td>NOSTATE</td>
</tr>
<tr>
<td>OS-EXT-STS:task_state</td>
<td>scheduling</td>
</tr>
<tr>
<td>OS-EXT-STS:vm_state</td>
<td>building</td>
</tr>
<tr>
<td>OS-SRV-USG:launched_at</td>
<td>None</td>
</tr>
<tr>
<td>OS-SRV-USG:terminated_at</td>
<td>None</td>
</tr>
<tr>
<td>accessIPv4</td>
<td></td>
</tr>
<tr>
<td>accessIPv6</td>
<td></td>
</tr>
<tr>
<td>addresses</td>
<td></td>
</tr>
<tr>
<td>adminPass</td>
<td>rvyh5M9VUt2R</td>
</tr>
<tr>
<td>config_drive</td>
<td></td>
</tr>
<tr>
<td>created</td>
<td>2017-02-17T09:14:48Z</td>
</tr>
<tr>
<td>flavor</td>
<td>m1.nano (0)</td>
</tr>
<tr>
<td>hostId</td>
<td></td>
</tr>
<tr>
<td>id</td>
<td>f8f27599-2733-4283-9043-a6451ebd9d9</td>
</tr>
<tr>
<td>image</td>
<td>cirros (6846e263-d0c9-46da-b643-4e95340dddef8)</td>
</tr>
<tr>
<td>key_name</td>
<td>None</td>
</tr>
<tr>
<td>name</td>
<td>host3</td>
</tr>
<tr>
<td>os-extended-volumes:volumes_attached</td>
<td>[]</td>
</tr>
<tr>
<td>progress</td>
<td>0</td>
</tr>
<tr>
<td>project_id</td>
<td>f5a2b881391e4170b1649c7343e0b361</td>
</tr>
<tr>
<td>properties</td>
<td></td>
</tr>
<tr>
<td>security_groups</td>
<td>[{u'name': u'default'}]</td>
</tr>
<tr>
<td>status</td>
<td>BUILD</td>
</tr>
<tr>
<td>updated</td>
<td>2017-02-17T09:14:49Z</td>
</tr>
<tr>
<td>user_id</td>
<td>4bc1f71e027348a6b81ab62f93bbc9d8</td>
</tr>
</tbody>
</table>
```
Creating and launching instance host4 with:

- **Flavour:** m1.nano
- **Image:** cirros
- **Network UUID:** 18a99a37-c0bd-4d8a-bdcd-9fad01c267a9
- **Security group:** default

+--------------------------------------+-----------------------------------------------+
<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS-DCF:diskConfig</td>
<td>MANUAL</td>
</tr>
<tr>
<td>OS-EXT-AZ:availability_zone</td>
<td></td>
</tr>
<tr>
<td>OS-EXT-STS:power_state</td>
<td>NOSTATE</td>
</tr>
<tr>
<td>OS-EXT-STS:task_state</td>
<td>scheduling</td>
</tr>
<tr>
<td>OS-EXT-STS:vm_state</td>
<td>building</td>
</tr>
<tr>
<td>OS-SRV-USG:launched_at</td>
<td>None</td>
</tr>
<tr>
<td>OS-SRV-USG:terminated_at</td>
<td>None</td>
</tr>
<tr>
<td>accessIPv4</td>
<td></td>
</tr>
<tr>
<td>accessIPv6</td>
<td></td>
</tr>
<tr>
<td>addresses</td>
<td></td>
</tr>
<tr>
<td>adminPass</td>
<td>m6aFFtP5FULK</td>
</tr>
<tr>
<td>config_drive</td>
<td></td>
</tr>
<tr>
<td>created</td>
<td>2017-02-17T09:14:53Z</td>
</tr>
<tr>
<td>flavor</td>
<td>m1.nano (0)</td>
</tr>
<tr>
<td>hostId</td>
<td></td>
</tr>
<tr>
<td>id</td>
<td>2c3fbb81-d3b4-4288-9275-31cce7aa5216</td>
</tr>
<tr>
<td>image</td>
<td>cirros (6846e263-d0c9-46da-b643-4e95340ddef8)</td>
</tr>
<tr>
<td>key_name</td>
<td>None</td>
</tr>
<tr>
<td>name</td>
<td>host4</td>
</tr>
<tr>
<td>os-extended-volumes:volumes_attached</td>
<td>[]</td>
</tr>
<tr>
<td>progress</td>
<td>0</td>
</tr>
<tr>
<td>project_id</td>
<td>f5a2b881391e4170b1649c7343e0b361</td>
</tr>
<tr>
<td>properties</td>
<td></td>
</tr>
<tr>
<td>security_groups</td>
<td>[{u'name': u'default'}]</td>
</tr>
<tr>
<td>status</td>
<td>BUILD</td>
</tr>
<tr>
<td>updated</td>
<td>2017-02-17T09:14:54Z</td>
</tr>
<tr>
<td>user_id</td>
<td>4bc1f7f7e027348a6b81ab62f93bfc9df</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>-----------------------------------------------</td>
</tr>
</tbody>
</table>

**Server list**

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Status</th>
<th>Networks</th>
<th>Image Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>2c3fbb81-d3b4-4288-9275-31cce7aa5216</td>
<td>host4</td>
<td>BUILD</td>
<td></td>
<td>cirros</td>
</tr>
<tr>
<td>f8f27599-2733-4283-9043-a4451bcb9df4</td>
<td>host3</td>
<td>BUILD</td>
<td></td>
<td>cirros</td>
</tr>
<tr>
<td>da5f338-296e-4053-b9f0-bc6718f0d1b</td>
<td>host2</td>
<td>BUILD</td>
<td></td>
<td>cirros</td>
</tr>
<tr>
<td>0905d664-353a-4f92-bfc2-4f4630b69564</td>
<td>host1</td>
<td>ACTIVE</td>
<td>provider=203.0.113.109</td>
<td>cirros</td>
</tr>
</tbody>
</table>

Create Router: router1

+--------------------------------------|--------------------------------------+
<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>admin_state_up</td>
<td>UP</td>
</tr>
<tr>
<td>availability_zone_hints</td>
<td></td>
</tr>
<tr>
<td>availability_zones</td>
<td></td>
</tr>
<tr>
<td>created_at</td>
<td>2017-02-17T09:15:00Z</td>
</tr>
<tr>
<td>description</td>
<td></td>
</tr>
<tr>
<td>external_gateway_info</td>
<td>null</td>
</tr>
<tr>
<td>flavor_id</td>
<td>None</td>
</tr>
<tr>
<td>headers</td>
<td></td>
</tr>
<tr>
<td>id</td>
<td>c405742c-7274-4d8e-8343-e5a03751903b</td>
</tr>
<tr>
<td>name</td>
<td>router1</td>
</tr>
<tr>
<td>project_id</td>
<td>f5a2b881391e4170b1649c7343e0b361</td>
</tr>
<tr>
<td>project_id</td>
<td>f5a2b881391e4170b1649c7343e0b361</td>
</tr>
<tr>
<td>revision_number</td>
<td>2</td>
</tr>
<tr>
<td>routes</td>
<td></td>
</tr>
<tr>
<td>status</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>updated_at</td>
<td>2017-02-17T09:15:00Z</td>
</tr>
</tbody>
</table>

Set gateway for router router1

Adding PRIV-SUBNET to router1
Adding default route to router1 via 203.0.113.1

Router: router1 configuration

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>admin_state_up</td>
<td>UP</td>
</tr>
<tr>
<td>availability_zone_hints</td>
<td></td>
</tr>
<tr>
<td>availability_zones</td>
<td>nova</td>
</tr>
<tr>
<td>created_at</td>
<td>2017-02-17T09:15:00Z</td>
</tr>
<tr>
<td>description</td>
<td></td>
</tr>
<tr>
<td>external_gateway_info</td>
<td>{'network_id': &quot;1ad8799b-8d9a-4ddd-801f-942da3549ee4&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;enable_snat&quot;: true, &quot;external_fixed_ips&quot;: [{'subnet_id':</td>
</tr>
<tr>
<td></td>
<td>&quot;d0fa8e34-cedd-4574-934d-824bb92bcc97&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;ip_address&quot;: &quot;203.0.113.112&quot;}] }</td>
</tr>
<tr>
<td>flavor_id</td>
<td>None</td>
</tr>
<tr>
<td>id</td>
<td>c405742c-7274-4d8e-8343-e5a03751903b</td>
</tr>
<tr>
<td>name</td>
<td>router1</td>
</tr>
<tr>
<td>project_id</td>
<td>f5a2b881391e4170b1649c7343e0b361</td>
</tr>
<tr>
<td>project_id</td>
<td>f5a2b881391e4170b1649c7343e0b361</td>
</tr>
<tr>
<td>revision_number</td>
<td>7</td>
</tr>
<tr>
<td>routes</td>
<td>destination='0.0.0.0/0', gateway='203.0.113.1'</td>
</tr>
<tr>
<td>status</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>updated_at</td>
<td>2017-02-17T09:15:25Z</td>
</tr>
</tbody>
</table>
17. Using HEAT orchestration

17.1 Introduction

In section 8.8 the Heat Orchestration service was briefly described. Heat provides a template-based orchestration for describing a cloud application by running OpenStack API calls to generate running cloud applications. To do this it uses Heat Orchestration Templates (HOT). These templates define multiple composite cloud applications and when passed to the heat-api, they are interpreted and passed to the heat-engine. The heat-engine creates jobs that are passed to the core services to create the cloud storage, network and VM instances as defined within the template. Heat has a second API called the heat-api-cfn which allows it to interpret AWS CloudFormation templates also.
17.2 HEAT Orchestration Templates (HOT)

HOT uses YAML Ain't Markup Language (YAML) which is an easily readable data serialisation language that is commonly used for configuration files. Each HOT template in YAML follows this format.

```yaml
heat_template_version: 2016-10-14

description:
  # a description of the template

parameter_groups:
  # a declaration of input parameter groups and order

parameters:
  # declaration of input parameters

resources:
  # declaration of template resources

outputs:
  # declaration of output parameters

conditions:
  # declaration of conditions
```

17.2.1 Template version

The `heat_template_version` tells heat the format of the template as well as the features supported. From the Newton release, the version can be either the date of the heat release or the code name of the heat release.

- 2013-05-23
- 2014-10-16
- 2015-04-30
- 2015-10-15
- 2016-04-08
- 2016-10-14 | newton
- 2017-02-24 | ocata

17.2.2 Description:

This section provides an optional `description` of the template.

17.2.3 Parameter groups

The `parameter_groups` section is used to specify how input parameters should be grouped and the order to provide the parameters in.

```yaml
parameter_groups:
  - label: <human-readable label of parameter group>
    description: <description of the parameter group>
    parameters:
      - <param name>
      - <param name>
```
17.2.4 Parameters

This section specifies input parameters that have to be provided when instantiating the template.

parameters:
  <param name>:
    type: <string | number | json | comma_delimited_list | boolean>
    label: <human-readable name of the parameter>
    description: <description of the parameter>
    default: <default value for parameter>
    hidden: <true | false>
    constraints: <parameter constraints>
    immutable: <true | false>

17.2.5 Resources

The Resources section defines the resources that make up a stack deployed from the template. Each resource is defined as a separate block in the resources section with the following syntax

resources:
  <resource ID>:
    type: <resource type>
    properties:
      <property name>: <property value>
    metadata:
      <resource specific metadata>
    depends_on: <resource ID or list of ID>
    update_policy: <update policy>
    deletion_policy: <deletion policy>
    external_id: <external resource ID>
    condition: <condition name or expression or boolean>

17.2.6 Outputs

This section defines output parameters that should be available to the user after a stack has been created. Each output parameter is defined as a separate block.

outputs:
  <parameter name>:
    description: <description>
    value: <parameter value>
    condition: <condition name or expression or boolean>

17.2.7 Conditions

This section defines one or more conditions which are evaluated based on input parameter values provided when a user creates or updates a stack. For example, based on the result of a condition, user can conditionally create resources, user can conditionally set different values of properties, and user can conditionally give outputs of a stack.

conditions:
  <condition name1>: {expression1}
  <condition name2>: {expression2}
  ...

17.3 Creating single servers

Check if the heat service is operational. The command should return empty line to indicate there is no existing stack.

```
osbash@controller:~$ . demo-openrc.sh
osbash@controller:~$ openstack stack list
```

Consider the available flavours, images and security groups. Their names will be required when creating the server template.

```
osbash@controller:~$ openstack flavor list
```

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>RAM</th>
<th>Disk</th>
<th>Ephemeral</th>
<th>VCPUs</th>
<th>Is Public</th>
</tr>
</thead>
<tbody>
<tr>
<td>f19dab3a-9909-406d-a3fb-9d48fc7a518f</td>
<td>m1.nano</td>
<td>1024</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>True</td>
</tr>
</tbody>
</table>

If it doesn't already exist add a new flavour. Note that the flavour must be added as the administrator user.

```
osbash@controller:~$ . admin-openrc.sh
osbash@controller:~$ openstack flavor create --vcpus 1 --ram 512 --disk 1 ml.nano
```

```
osbash@controller:~$ . demo-openrc.sh
```

If not then create the rules within the default security group.

```
osbash@controller:~$ openstack security group rule create --proto tcp --dst-port 22 default
osbash@controller:~$ openstack security group rule create --proto icmp default
```

Create a YAML template. This template specifies the flavour, image and public network (pub_net). These parameters are pulled together under resources declare based on the parameters selects what needs to be instantiated. The outputs section specifies output parameters available to users once the template has been instantiated. This is optional and can be omitted when no output values are required.
osbash@controller:~$ cat <<'EOM' > ~/Server.yaml

# This is a hello world HOT template just defining a single compute server.
#
heat_template_version: 2016-04-08

description: Hello world HOT template defining a single server.

parameters:

  flavor:
    type: string
    description: Flavour for the server to be created
    default: m1.nano
    constraints:
      - custom_constraint: nova.flavor

  image:
    type: string
    description: Image name
    default: cirros
    constraints:
      - custom_constraint: glance.image

  pub_net:
    type: string
    description: ID of public network
    default: provider
    constraints:
      - custom_constraint: neutron.network

resources:

  server:
    type: OS::Nova::Server
    properties:
      image: { get_param: image }
      flavor: { get_param: flavor }
      networks:
        - network: { get_param: pub_net }

outputs:

  server_networks:
    description: The networks of the deployed server
    value: { get_attr: [server, networks] }

EOM
This stack is created using the *defaults* within the YAML file. It causes the defined server to be instantiated.

```
osbash@controller:~$ openstack stack create --template Server.yaml singlestack
```

```
+---------------------+----------------------------------------------------+
| Field               | Value                                              |
+---------------------+----------------------------------------------------+
| id                  | 04769c30-590f-4354-9c68-04158407a283               |
| stack_name          | singlestack                                        |
| description         | Hello world HOT template defining a single server. |
| creation_time       | 2017-02-17T13:23:43Z                               |
| updated_time        | None                                               |
| stack_status        | CREATE_IN_PROGRESS                                 |
| stack_status_reason | Stack CREATE started                               |
+---------------------+----------------------------------------------------+

Review the actions the *heat-engine* push to the core services once the template has been interpreted.

```
osbash@controller:~$ openstack stack event list singlestack
2017-02-17 13:23:44Z [singlestack]: CREATE_IN_PROGRESS  Stack CREATE started
2017-02-17 13:23:44Z [server]: CREATE_IN_PROGRESS  state changed
2017-02-17 13:24:02Z [server]: CREATE_COMPLETE  state changed
2017-02-17 13:24:02Z [singlestack]: CREATE_COMPLETE  Stack CREATE completed successfully
```
osbash@controller:~$ openstack server list

+---------------------------------+---------------------------------+---------------------------------+---------------------------------+---------------------------------+---------------------------------+---------------------------------+---------------------------------+
| ID                             | Name                            | Status                          | Networks                       | Provider                        | Image Name                      | Image ID                        |
|---------------------------------+---------------------------------+---------------------------------+---------------------------------+---------------------------------+---------------------------------+---------------------------------|
| 40d4d8e5-3c-4a88-b0c1-15402baa35 | single-stack-server-1npuyg1638s3 | ACTIVE                          |                                |                                |                                |                                |
|                                |                                 |                                |                                |                                |                                |                                |
|                                |                                 |                                |                                |                                |                                |                                |
It is possible to change parameters from the default by specifying them as part of the command.

```
$ openstack stack create --template Server.yaml --parameter flavor=m1.small secondstack
```

+---------------------+----------------------------------------------------+
| Field               | Value                                              |
| id                  | 77dd9180-1472-4487-b909-ce19f2af5c0b               |
| stack_name          | secondstack                                        |
| description         | Hello world HOT template defining a single server. |
| creation_time       | 2017-02-17T13:26:30Z                               |
| updated_time        | None                                               |
| stack_status        | CREATE_IN_PROGRESS                                 |
| stack_status_reason | Stack CREATE started                               |
+---------------------+----------------------------------------------------+

```
$ openstack stack event list secondstack
```

2017-02-17 13:26:30Z [secondstack]: CREATE_IN_PROGRESS Stack CREATE started
2017-02-17 13:26:31Z [server]: CREATE_IN_PROGRESS state changed
2017-02-17 13:26:45Z [server]: CREATE_COMPLETE state changed
2017-02-17 13:26:45Z [secondstack]: CREATE_COMPLETE Stack CREATE completed successfully
osbash@controller:-$ openstack server list

+-----------------+----------+----------------+---------+---------+---------+---------------+
| ID              | Name     | Image          | Status  | Network | Provider | Created       |
+-----------------+----------+----------------+---------+---------+---------+---------------+
| c7be5a2c2313 - 4e1-b47-5d10-7a9a55 | secondstack server - psdjon5kvh | cirsos | ACTIVE  | OpenStack | 203.0.113.109 | 2016-12-02 17:26:35 |
| 4e3a8263c7 - 48-5b2c-5134-20a303 | singlestack server - epnplig16-show | cirsos | ACTIVE  | OpenStack | 203.0.113.113 | 2016-12-02 17:26:35 |
17.4 Create complete network and servers

Reviewing the "Scripting the operation" on page 151 where a new private network was created and servers allocated to each. This section looks at how the same can be achieved using Heat orchestration.

To simplify matters a parent YAML file will be used which will create the servers. It will also call on a child YAML file to build the networks.

17.4.1 Networks – child template

**Parameters**

This YAML file starts with a parameter section describing the public network (pub_net) as the existing provider. It then defines various attributes required to establish the private network (pri_net) and the associate private network subnet (pri_subnet).

**Resources**

The resources section defines the pri_net as a network and generates the associated subnet pri_subnet by calling on parameters from the section above.

A router is created whose external gateway information is also extracted from the parameters section, i.e. pub_net pointing to the provider network. An additional interface is added to the router and the pri_subnet is associated with it.

**Outputs**

The outputs section returns the names of the networks as key/value pairs:

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>pub_net_name</td>
<td>provider</td>
</tr>
<tr>
<td>pri_net_name</td>
<td>Extract the name given at create time</td>
</tr>
<tr>
<td>router_gw</td>
<td>Extract the External Gateway information</td>
</tr>
</tbody>
</table>

If this template is executed on its own then these values can be viewed with the command:

`openstack stack show <stackname>`

However if this template is called from another then the values are passed back to the parent template.
osbash@controller:~$ cat <<'EOM' > ~/networks.yaml
heat_template_version: 2016-04-08

description: Template that creates a private network.

parameters:

  pub_net:
    type: string
    label: Public network name or ID
    description: Public network with floating IP addresses.
    default: provider

  pri_net_cidr:
    type: string
    default: '192.168.95.0/24'
    description: Private network address (CIDR notation)

  pri_net_gateway:
    type: string
    default: '192.168.95.1'
    description: Private network gateway address

  pri_net_nameserver:
    type: comma_delimited_list
    default: '8.8.8.8'
    description: Private network DNS Server address

  pri_net_enable_dhcp:
    type: boolean
    default: 'True'
    description: enable DHCP Server

  pri_net_pool_start:
    type: string
    default: '192.168.95.10'
    description: Private network Start IP address allocation pool

  pri_net_pool_end:
    type: string
    default: '192.168.95.20'
    description: Private network End IP address allocation pool

  pri_net_nexthop:
    type: string
    default: '203.0.113.1'
    description: nexthop address for default route

resources:

  pri_net:
    type: OS::Neutron::Net

  pri_subnet:
    type: OS::Neutron::Subnet
    properties:
      network_id: { get_resource: pri_net }
      cidr: { get_param: pri_net_cidr }
      dns_nameservers: { get_param: pri_net_nameserver }
      gateway_ip: { get_param: pri_net_gateway }
      enable_dhcp: { get_param: pri_net_enable_dhcp }
      allocation_pools:
        - start: { get_param: pri_net_pool_start }
        - end: { get_param: pri_net_pool_end }

host_routes:
    - destination: '0.0.0.0/0'
      nexthop: { get_param: pri_net_nexthop }

router:
    type: OS::Neutron::Router
    properties:
        external_gateway_info:
            network: { get_param: pub_net }

router-interface:
    type: OS::Neutron::RouterInterface
    properties:
        router_id: { get_resource: router }
        subnet: { get_resource: pri_subnet }

outputs:

    pub_net_name:
        description: The public network.
        value: provider

    pri_net_name:
        description: The private network.
        value: { get_attr: [pri_net, name] }

    router_gw:
        description: Router gateway information
        value: { get_attr: [router, external_gateway_info] }

EOM

To prove the network template it is possible to run it on its own before working on the parent. This demonstrates that to this point everything is operational.

```
osbash@controller:~$ . demo-openrc.sh

osbash@controller:~$ openstack stack create --template networks.yaml netstack
```

```
+---------------------+------------------------------------------+
| Field               | Value                                    |
+---------------------+------------------------------------------+
| id                  | 76c0688f-bed5-44fb-b49b-9da1db0c5cd3     |
| stack_name          | netstack                                 |
| description         | Template that creates a private network. |
| creation_time       | 2017-02-22T13:26:31Z                     |
| updated_time        | None                                     |
| stack_status        | CREATE_IN_PROGRESS                       |
| stack_status_reason | Stack CREATE started                     |
+---------------------+------------------------------------------+
```
osbash@controller:~$ openstack stack show netstack

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>76c0688f-bed5-44fb-b49b-9da1db0c5cd3</td>
</tr>
<tr>
<td>stack_name</td>
<td>netstack</td>
</tr>
<tr>
<td>description</td>
<td>Template that creates a private network.</td>
</tr>
<tr>
<td>creation_time</td>
<td>2017-02-22T13:26:31Z</td>
</tr>
<tr>
<td>stack_status</td>
<td>CREATE_COMPLETE</td>
</tr>
<tr>
<td>stack_status_reason</td>
<td>Stack CREATE completed successfully</td>
</tr>
<tr>
<td>parameters</td>
<td>OS::project_id: bdd928b9d2e94a67ad927bc98611917c</td>
</tr>
<tr>
<td></td>
<td>OS::stack_id: 76c0688f-bed5-44fb-b49b-9da1db0c5cd3</td>
</tr>
<tr>
<td></td>
<td>OS::stack_name: netstack</td>
</tr>
<tr>
<td></td>
<td>pri_net_cidr: 192.168.95.0/24</td>
</tr>
<tr>
<td></td>
<td>pri_net_enable_dhcp: 'True'</td>
</tr>
<tr>
<td></td>
<td>pri_net_gateway: 192.168.95.1</td>
</tr>
<tr>
<td></td>
<td>pri_net_nameserver: '[u''8.8.8.8'']'</td>
</tr>
<tr>
<td></td>
<td>pri_net_nexthop: 203.0.113.1</td>
</tr>
<tr>
<td></td>
<td>pri_net_pool_end: 192.168.95.20</td>
</tr>
<tr>
<td></td>
<td>pri_net_pool_start: 192.168.95.10</td>
</tr>
<tr>
<td></td>
<td>pub_net: provider</td>
</tr>
<tr>
<td>outputs</td>
<td>- description: The public network.</td>
</tr>
<tr>
<td></td>
<td>output_key: pub_net_name</td>
</tr>
<tr>
<td></td>
<td>output_value: provider</td>
</tr>
<tr>
<td></td>
<td>- description: Router gateway information</td>
</tr>
<tr>
<td></td>
<td>output_key: router_gw</td>
</tr>
<tr>
<td></td>
<td>output_value:</td>
</tr>
<tr>
<td></td>
<td>enable_snat: true</td>
</tr>
<tr>
<td></td>
<td>external_fixed_ips:</td>
</tr>
<tr>
<td></td>
<td>- ip_address: 203.0.113.112</td>
</tr>
<tr>
<td></td>
<td>subnet_id: c52e0181-5431-4be4-8b0d-e76b15750d77</td>
</tr>
<tr>
<td></td>
<td>network_id: 785f50d2-6690-4e0b-99b3-530741eb1d76</td>
</tr>
<tr>
<td></td>
<td>- description: The private network.</td>
</tr>
<tr>
<td></td>
<td>output_key: pri_net_name</td>
</tr>
<tr>
<td></td>
<td>output_value: netstack-pri_net-lrx3l746npza</td>
</tr>
<tr>
<td>links</td>
<td>- href: <a href="http://controller:8004/v1/bdd928b9d2e94a67ad927bc98611917c/stacks/netstack/76c0688f-bed5-44fb-b49b-9da1db0c5cd3">http://controller:8004/v1/bdd928b9d2e94a67ad927bc98611917c/stacks/netstack/76c0688f-bed5-44fb-b49b-9da1db0c5cd3</a> rel: self</td>
</tr>
<tr>
<td>parent</td>
<td>None</td>
</tr>
<tr>
<td>disable_rollback</td>
<td>True</td>
</tr>
<tr>
<td>deletion_time</td>
<td>None</td>
</tr>
<tr>
<td>stack_user_project_id</td>
<td>ec28c0b118914749b3f8d4d3c866a03</td>
</tr>
<tr>
<td>capabilities</td>
<td>[]</td>
</tr>
<tr>
<td>notification_topics</td>
<td>[]</td>
</tr>
<tr>
<td>stack_owner</td>
<td>None</td>
</tr>
<tr>
<td>timeout_mins</td>
<td>None</td>
</tr>
<tr>
<td>tags</td>
<td>null</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

Note the outputs. Shortly it will become clear that these values are passed to the parent YAML template.

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>pub_net_name</td>
<td>provider</td>
</tr>
<tr>
<td>pri_net_name</td>
<td>netstack-pri_net-lrx3l746npza</td>
</tr>
<tr>
<td>router_gw</td>
<td>ip_address: 203.0.113.112</td>
</tr>
</tbody>
</table>
Review the network in horizon.

Network > Network Topology > Topology : Normal

Illustration 29: Network topology

Network > Network Topology > Graph : Toggle labels

Illustration 30: Network graph

17.4.2 Delete stack

Delete the stack netstack before continuing. It was just a test.

osbash@controller:~$ openstack stack delete netstack
Are you sure you want to delete this stack(s) [y/N]? y
17.4.3 Parent template

**Parameters**

This YAML file parameter section describes the flavour and image that will be used to create the hosts.

**Resources**

The resources section defines networks by calling on the networks.yaml template. The outputs seen earlier from that child template are fed back to the parent template. pub_net_name and pri_net_name are called during the development of the host resources to give up the values from the execution of the child template.

**Outputs**

The outputs section returns the networks of each of the hosts as well as the external_gateway_info that was gathered by the child template and pass as the key router_gw.

```bash
osbash@controller:~$ cat <<'EOM' > ~/servers_networks.yaml
heat_template_version: 2016-04-08
description: Template that creates hosts connected to two networks
parameters:
  image:
    type: string
    label: Image name or ID
    description: Image to be used for server.
    default: cirros
  flavor:
    type: string
    label: Flavor
    description: Type of instance (flavor) for the compute instance.
    default: m1.nano
resources:
  networks:
    type: networks.yaml
  host1:
    type: OS::Nova::Server
    properties:
      image: { get_param: image }  
      flavor: { get_param: flavor }  
      networks:
        - network: { get_attr: [networks, pub_net_name] }  
```
host2:
  type: OS::Nova::Server
  properties:
    image: { get_param: image }
    flavor: { get_param: flavor }
    networks:
      - network: { get_attr: [networks, pub_net_name] }

host3:
  type: OS::Nova::Server
  properties:
    image: { get_param: image }
    flavor: { get_param: flavor }
    networks:
      - network: { get_attr: [networks, pri_net_name] }

host4:
  type: OS::Nova::Server
  properties:
    image: { get_param: image }
    flavor: { get_param: flavor }
    networks:
      - network: { get_attr: [networks, pri_net_name] }

outputs:

host1_networks:
  description: The networks of the deployed server
  value: { get_attr: [host1, networks] }

host2_networks:
  description: The networks of the deployed server
  value: { get_attr: [host2, networks] }

host3_networks:
  description: The networks of the deployed server
  value: { get_attr: [host3, networks] }

host4_networks:
  description: The networks of the deployed server
  value: { get_attr: [host4, networks] }

router_gateway:
  description: The router gateway information
  value: { get_attr: [networks, router_gw] }

EOM

osbash@controller:~$ openstack stack create --template servers_networks.yaml fullstack

+---------------------+-------------------------------------------------------+
| Field       | Value                                                 |
+---------------------+-------------------------------------------------------+
| id          | 53b3f882-1a3f-4b1f-b838-f0e5cbf61002                  |
| stack_name  | fullstack | Template that creates hosts connected to two networks |
| creation_time| 2017-02-22T13:49:06Z                                  |
| updated_time| None       |
| stack_status| CREATE_IN_PROGRESS                                   |
| stack_status_reason| Stack CREATE started          |
Note the External IP address. A route will need to be made to the 192.168.95.0/24 network via this IP address on the hypervisor.
17.4.4 Stack events

Review the actions the heat-engine push to the core services once the template has been interpreted.

```
osbash@controller:~$ openstack stack event list fullstack
2017-02-22 13:49:07Z [fullstack]: CREATE_IN_PROGRESS  Stack CREATE started
2017-02-22 13:49:07Z [networks]: CREATE_IN_PROGRESS  state changed
2017-02-22 13:49:19Z [networks]: CREATE_COMPLETE  state changed
2017-02-22 13:49:22Z [host4]: CREATE_IN_PROGRESS  state changed
2017-02-22 13:49:23Z [host3]: CREATE_IN_PROGRESS  state changed
2017-02-22 13:49:25Z [host2]: CREATE_IN_PROGRESS  state changed
2017-02-22 13:49:26Z [host1]: CREATE_IN_PROGRESS  state changed
2017-02-22 13:51:08Z [host1]: CREATE_COMPLETE  state changed
2017-02-22 13:51:08Z [host4]: CREATE_COMPLETE  state changed
2017-02-22 13:51:15Z [host2]: CREATE_COMPLETE  state changed
2017-02-22 13:51:24Z [host3]: CREATE_COMPLETE  state changed
2017-02-22 13:51:24Z [fullstack]: CREATE_COMPLETE  Stack CREATE completed successfully
```
Review the servers and router created.

```
osbash@controller:~$ openstack server list
osbash@controller:~$ openstack router list
osbash@controller:~$ openstack network list
```
17.4.5 Add a route on the hypervisor to the private network

Add a static route from the hypervisor host to the private network. First determine the IP address assigned to the OpenStack router provider interface. This can be done with the last command or

```bash
osbash@controller:~$ openstack router list --column Name
+------------------------------------------------------+
| Name                                                 |
+------------------------------------------------------+
| fullstack-networks-zj23gzi6zv3n-router-c2brepuddtje |
+------------------------------------------------------+
```

```bash
osbash@controller:~$ openstack router show fullstack-networks-zj23gzi6zv3n-router-c2brepuddtje | grep external_gateway_info | awk -F "" '{print $16}'
203.0.113.102
```

Then add the static route to the private network.

```bash
ada:~$ sudo ip route add 192.168.95.0/24 metric 1 nexthop via 203.0.113.102
```
17.4.6 Test the configuration

From the hypervisor host ping the four hosts.

```bash
ada:~$ ping -c1 203.0.113.103
PING 203.0.113.103 (203.0.113.103) 56(84) bytes of data.
64 bytes from 203.0.113.103: icmp_seq=1 ttl=64 time=1.79 ms

--- 203.0.113.103 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.791/1.791/1.791/0.000 ms

ada:~$ ping -c1 203.0.113.112
PING 203.0.113.112 (203.0.113.112) 56(84) bytes of data.
64 bytes from 203.0.113.112: icmp_seq=1 ttl=64 time=1.58 ms

--- 203.0.113.112 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.580/1.580/1.580/0.000 ms

ada:~$ ping -c1 192.168.95.13
PING 192.168.95.13 (192.168.95.13) 56(84) bytes of data.
64 bytes from 192.168.95.13: icmp_seq=1 ttl=63 time=3.48 ms

--- 192.168.95.13 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 3.485/3.485/3.485/0.000 ms

ada:~$ ping -c1 195.168.95.20
PING 192.168.95.20 (192.168.95.20) 56(84) bytes of data.
64 bytes from 192.168.95.20: icmp_seq=1 ttl=63 time=3.67 ms

--- 192.168.95.20 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 3.670/3.670/3.670/0.000 ms
```

Connect to one of the hosts on the private network and confirm connectivity to the Internet.

```bash
ada:~$ ssh cirros@192.168.95.13
cirros@192.168.95.13's password: cubswin:)

$ ping -c1 8.8.8.8
PING 8.8.8.8 (8.8.8.8): 56 data bytes
64 bytes from 8.8.8.8: seq=0 ttl=54 time=284.784 ms

--- 8.8.8.8 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 284.784/284.784/284.784 ms
```
17.4.7 Review topology on the Horizon dashboard

Login as user demo password: demo_user_pass and select:

Network > Network Topology > Topology : Normal

Illustration 31: Network topology

Network > Network Topology > Graph : Toggle labels

Illustration 32: Network graph

So using Heat orchestration a network with hosts can be build that is for all intent and purpose identical to that created in Chapter 16 - Creating networks.
18. Appendices

18.1 Appendix 1 - NAT Masquerade script for Hypervisor host

Enable IP forwarding and setup masquerade in IP Tables for Linux netfilter. `enp0s3` is the interface on the hypervisor host that connects to the Internet, it is considered the outside network for the NAT masquerade. (note if this computer is connected by wireless it is likely that this interface will actually be `wlp4s0`). On KVM/QEMU the provider network is typically `virbr2`, while on VirtualBox the network is typically `vboxnet1` with the IP addresses for both from the 203.0.113.0/24 network. It is from this network that instances are assigned IP addresses from a pool. This address pool is the inside network for the purpose of the NAT masquerade.

```
ada:~$ cat <<'EOM' > $OS_LAB/nat_tables.sh
#!/bin/bash

# NAT masquerade rules for hypervisor, hosting OpenStack testbed

INTERFACE=enp3s0     # Unhash for wired Ethernet interface
#INTERFACE=wlp4s0    # Unhash for wireless WIFI interface

# Select instance private network
NETWORK=virbr2         # For KVM/QEMU
#NETWORK=vboxnet1      # For VirtualBox

# Flush iptables
iptables -F
iptables -F -t nat

# Enable IP forwarding
echo "echo "1" > /proc/sys/net/ipv4/ip_forward"
"1" > /proc/sys/net/ipv4/ip_forward

# Load GNU/Linux kernel modules
modprobe ip_tables
modprobe ip_conntrack

# Add IPTABLES rules
iptables -t nat -A POSTROUTING -- $INTERFACE -j MASQUERADE
iptables -A FORWARD -i $INTERFACE -o $NETWORK --state RELATED,ESTABLISHED -j ACCEPT
iptables -A FORWARD -i $NETWORK -- $INTERFACE -j ACCEPT

# Print iptables
iptables -t nat -L POSTROUTING

# END
EOM

ada:~$ chmod +x $OS_LAB/nat_tables.sh
```
### 18.2 Appendix 2 – Cluster Start/Stop script

This script acts as a start/stop script for the cluster on the host.

```bash
ads:~$ cat <<'EOM' > ~/start-stop-cluster.sh
#!/bin/bash

###########################################
# program: start-stop-cluster.sh         #
# Author: Diarmuid O’Briain              #
# Copyright ©2017 C²S Consulting         #
# License: www.gnu.org/licenses/gpl.txt  #
###########################################

PROVIDER=''
if [[ `echo "$0" | grep './'` ]]; then
  command=`echo "$0"|awk -F '/' '{print $2}'`
else
  command=$0
fi

# Help function
function usage {
  echo -e "usage: $command <PROVIDER> <START | STOP>  help, -h, --help, --help
  PROVIDER:: kvm | vbox
  echo "$0" awk -F '/' '{print $2}'
  echo "$0" kvm = Kernel based Virtual Machine/Quick Emulator (KVM/QEMU)
  echo "$0" vbox = Oracle VirtualBox
  echo "$0" Start or Stop the Virtual Machines in the cluster"
  exit
}

# Arguments from the command line
if [[ $# -lt 1 ]]; then # Deal with too few arguments
  echo -e "\nNot enough arguments\n"
  usage
elif [[ $# -gt 2 ]]; then # Deal with too many arguments
  echo -e "\nToo many arguments\n"
  usage
elif [[ $1 =~ (kvm|vbox) ]]; then # Deal with request for help
  echo -e "\nNot an acceptable option\n"
  usage
else
  echo -e "\nSelected provider is: $PROVIDER and the cluster will $ACTION\n"

# Action nodes
if [[ $PROVIDER =~ 'kvm' ]]; then
  if ! virsh -c 'qemu:///system' net-list | egrep 'labs-mgmt|labs-provider'; then
    virsh -c 'qemu:///system' net-start 'labs-mgmt'
  fi
  sleep 5
  virsh -c 'qemu:///system' start 'controller'
  virsh -c 'qemu:///system' start 'compute1'
else
  if [[ $PROVIDER =~ 'vbox' ]]; then
    vboxmanage startvm 'controller' --type headless
    vboxmanage startvm 'compute1' --type headless
    echo
  fi
fi

# Show cluster
echo -e "\nCluster state\n"
```

---

**OpenStack Pike**

**02 Oct 2017**
if [[ $PROVIDER =~ 'kvm' ]] && [[ $ACTION =~ 'stop' ]]; then
    while true; do
        listout=$(virsh -c 'qemu:///system' list)
        if ! [[ $(echo $listout |egrep 'controller|compute1') ]]; then
            break
        fi
        echo -n '. ';sleep 2
    done
    echo
    virsh -c 'qemu:///system' net-list
    virsh -c 'qemu:///system' list
elif [[ $PROVIDER =~ 'kvm' ]] && [[ $ACTION =~ 'start' ]]; then
    virsh -c 'qemu:///system' net-list
    virsh -c 'qemu:///system' list
elif [[ $PROVIDER =~ 'vbox' ]] && [[ $ACTION =~ 'start' ]]; then
    echo 'Running VMs'
    vboxmanage list runningvms | egrep 'controller|computel'
else
    echo 'VMs in a shutdown state'
    vboxmanage list vms | egrep 'controller|computel'
fi
# END
EOM

chmod +x ~/start-stop-cluster.sh
18.2.1 Running for a KVM/QEMU system

`ada:~$ ./start-stop-cluster.sh kvm start`

Selected provider is: kvm and the cluster will start

Powering up KVM/QEMU nodes

Domain controller started

Domain compute1 started

Cluster state

```
<table>
<thead>
<tr>
<th>Name</th>
<th>State</th>
<th>Autostart</th>
<th>Persistent</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>active</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>labs-mgmt</td>
<td>active</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>labs-provider</td>
<td>active</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>29</td>
<td>controller</td>
<td>running</td>
</tr>
<tr>
<td>30</td>
<td>compute1</td>
<td>running</td>
</tr>
</tbody>
</table>
```

`ada:~$ ./start-stop-cluster.sh kvm stop`

Selected provider is: kvm and the cluster will stop

Powering down KVM/QEMU nodes

Domain controller is being shutdown

Domain compute1 is being shutdown

Cluster state

```
<table>
<thead>
<tr>
<th>Name</th>
<th>State</th>
<th>Autostart</th>
<th>Persistent</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>active</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>labs-mgmt</td>
<td>active</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>labs-provider</td>
<td>active</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

---

netLabs!UG

OpenStack Pike

02 Oct 2017
18.2.2 Running for a VirtualBox system

* ada:~$ ~/start-stop-cluster.sh vbox start

Selected provider is: vbox and the cluster will start

Powering up VirtualBox nodes

Waiting for VM "controller" to power on...
VM "controller" has been successfully started.
Waiting for VM "compute1" to power on...
VM "compute1" has been successfully started.

Cluster state

Running VMs
"controller" {85cc5cd8-3392-49bd-bac8-76c4a8bed317}
"compute1" {42d461ef-79cf-49a7-a6fd-5bcfcafcd87c}

* ada:~$ ~/start-stop-cluster.sh vbox stop

Selected provider is: vbox and the cluster will stop

Powering off VirtualBox nodes

0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%

Cluster state

VMs in a shutdown state
"controller" {85cc5cd8-3392-49bd-bac8-76c4a8bed317}
"compute1" {42d461ef-79cf-49a7-a6fd-5bcfcafcd87c}
18.3 Appendix 3 - Clean nodes script for Hypervisor host

This script returns the OpenStack lab to a clean state after it has been worked for a while. Simply select the provider to run.

```
ada:~$ cat <<'EOM' > $OS_LAB/clean_nodes.sh
#!/bin/bash

###########################################
#  program: clean_nodes.sh                #
#  Author: Diarmuid O'Briain              #
#  Copyright ©2017 C²S Consulting         #
#  License: www.gnu.org/licenses/gpl.txt  #
###########################################

PROVIDER='
if [[ 'echo "$0" | grep './' ]]; then
    command='echo "$0"|awk -F '/' '{print $2}'
else
    command=$0
fi

# Help function
function usage {
    echo -e "usage: $command <PROVIDER>   help, -h, -help, --help
       PROVIDER:: kvm | vbox
       kvm = Kernel based Virtual Machine/Quick Emulator (KVM/QEMU)
       vbox = Oracle VirtualBox
       Note: For KVM/QEMU this command must be ran as sudo\n"
    exit
}

# Arguments from the command line
if [[ $# -lt 1 ]]; then   # Deal with too few arguments
    echo -e "\nNot enough arguments\n"
    usage
elif [[ $# -gt 1 ]]; then   # Deal with too many arguments
    echo -e "\nToo many arguments\n"
    usage
elif [[ $1 =~ (-h|-help|--help|help) ]]; then   # Deal with request for help
    usage
elif [[ $1 =~ (kvm|vbox) ]]; then   # Deal with legit option
    PROVIDER=$1
    echo -e "\nSelected provider is: $PROVIDER\n"
else
    echo -e "\nNot an acceptable option\n"
    usage
fi

echo -e "\nRestoring nodes to clean state\n"

# Powering off nodes
if [[ $PROVIDER =~ 'kvm' ]]; then
    echo -e "Powering off KVM/QEMU nodes\n"
    virsh -c 'qemu:///system' shutdown 'controller'
    virsh -c 'qemu:///system' shutdown 'compute1'
else
    echo -e "Powering off VirtualBox nodes\n"
    vboxmanage controlvm 'controller' poweroff
    vboxmanage controlvm 'compute1' poweroff
fi

# Wait for nodes to power down
echo -e "\nWaiting for nodes to power down completely\n"

if [[ $PROVIDER =~ 'kvm' ]]; then
    while [[ $1 ]]; do
        CONTROLLER_STATE=`virsh -c 'qemu:///system' list --all | grep -e 'controller' | awk '{print $3, $4}'`
        COMPUTE_STATE=`virsh -c 'qemu:///system' list --all | grep -e 'compute' | awk '{print $3, $4}'`
        printf "."; sleep 2
        if [[ $CONTROLLER_STATE =~ 'shut off' && $COMPUTE_STATE =~ 'shut off' ]]; then
            echo -e "\nController node and Compute1 node are in a shut down state\n"
        fi
    done
fi
```


break
fi
done
else
while [[ 1 ]]; do
    CONTROLLER_STATE=`vboxmanage showvminfo 'controller' | grep '^State' | awk '{print $2}'`
    COMPUTE_STATE=`vboxmanage showvminfo 'controller' | grep '^State' | awk '{print $2}'`
    printf "."
    if [[ $CONTROLLER_STATE =~ 'powered' && $COMPUTE_STATE =~ 'powered' ]]; then
        echo -e "\n\nController node and Compute1 node are in a shut down state"
        break
    fi
fi

# Return nodes to last working snapshots
if [[ $PROVIDER =~ 'kvm' ]]; then
    echo -e "\nReverting KVM/QEMU nodes to earlier snapshots\n"
    virsh -c 'qemu:///system' snapshot-revert --domain 'controller' 
      --snapshotname 'controller_-_cluster_installed' --running
    virsh -c 'qemu:///system' snapshot-revert --domain 'compute1' 
      --snapshotname 'compute_-_cluster_installed' --running
else
    echo -e "\nReverting VirtualBox nodes to earlier snapshot\n"
    vboxmanage snapshot 'controller' restore 'controller_-_cluster_installed'
    echo
    vboxmanage snapshot 'compute1' restore 'compute_-_cluster_installed'
    echo
    vboxmanage startvm 'controller' --type headless
    echo
    vboxmanage startvm 'compute1' --type headless
    echo
fi

# Show clean nodes
if [[ $PROVIDER =~ 'kvm' ]]; then
    virsh -c 'qemu:///system' list
else
    vboxmanage list runningvms
    echo
fi

# END
EOM

ada:~$ chmod +x $OS_LAB/clean_nodes.sh
18.3.1 Running for a KVM/QEMU system

ada:~$ $OS_LAB/clean_nodes.sh kvm

Selected provider is: kvm

Restoring nodes to clean state
Powering off KVM/QEMU nodes
Domain controller is being shutdown
Domain compute1 is being shutdown

Waiting for nodes to power down completely

......

Controller node and Compute1 node are in a shut down state

Reverting KVM/QEMU nodes to earlier snapshot

Domain compute1 started

Clean running nodes

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>controller</td>
<td>running</td>
</tr>
<tr>
<td>8</td>
<td>Compute1</td>
<td>running</td>
</tr>
</tbody>
</table>
18.3.2 Running for a VirtualBox system

    ada:~$ $OS_LAB/clean_nodes.sh vbox

Selected provider is: vbox

Restoring nodes to clean state

Powering off VirtualBox nodes

0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%

Waiting for nodes to power down completely

Controller node and Compute1 node are in a shut down state

Reverting VirtualBox nodes to earlier snapshot

Restoring snapshot 3bbff8c3-8203-4226-8563-9fecd6e444b1
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%

Waiting for VM "controller" to power on...
VM "controller" has been successfully started.

Waiting for VM "compute1" to power on...
VM "compute1" has been successfully started.

Clean running nodes

"controller" {e18abd53-5c5c-4938-84af-ca4e6409a734}
"compute1" {bd283312-4d11-4e8f-9ab2-a08c91de59e3}
18.4 Appendix 4 - Script to launch a VM instance

This script if ran on the controller node after the OpenStack Labs install will create a VM instance with a 1 GB volume attached. Make sure to run `$OS_LAB/nat_tables.sh` on the hypervisor host to enable routing from the VM instances.

```bash
osbash@controller:~$ cat <<'EOM' > ~/instance_launch.sh
#!/bin/bash

Space: instance_launch.sh
# Author: Diarmuid O'Brien
# Copyright ©2017 C²S Consulting
# License: www.gnu.org/licenses/gpl.txt

# Run this script on the controller node
# Access the Controller node
# KVM/QEMU: ssh osbash@10.0.0.11
# VirtualBox: ssh -p 2230 osbash@localhost

# Make sure to run "sudo $OS_LAB/nat_tables.sh" on hypervisor host

# Variables

KEYNAME='mykey'
INSTANCE='cirroS-test'
VOLNAME='1GB-vol'
FLAVOUR='m1.nano'
IMAGE='cirros'
SSH_HOSTS_FILE='/home/osbash/.ssh/id_rsa'

echo; echo "Setting admin-openrc variables"
export OS_USERNAME=admin
export OS_PASSWORD=admin_user_secret
export OS_PROJECT_NAME=admin
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_DOMAIN_NAME=Default
export OS_AUTH_URL=http://10.0.0.11:35357/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2

echo; echo "Creating flavour $FLAVOUR"
openstack flavor create --id 0 --vcpus 1 --ram 64 --disk 1 $FLAVOUR

echo; echo "Setting demo-openrc variables"
export OS_USERNAME=demo
export OS_PASSWORD=demo_user_pass
export OS_PROJECT_NAME=demo
export OS_PROJECT_DOMAIN_NAME=Default
export OS_AUTH_URL=http://10.0.0.11:35357/v3

echo; echo "Restricting ~/$KEYNAME.pem access rights"
chmod 600 ~/$KEYNAME.pem

echo; echo "Adding port 22 (SSH) and ICMP to default security group"
openstack security group rule create --proto tcp --dst-port 22 default
openstack security group rule create --proto icmp default

NIC=$(openstack network list | grep provider | awk '{print $2}')

echo; echo "Extracting provider network UUID: $NIC"
```

```
echo; echo "Creating and launching instance $INSTANCE with:"

```
echo -e "\t\tFlavour: $FLAVOUR"
echo -e "\t\tImage: $IMAGE"
echo -e "\t\tNetwork UUID=$NIC"
echo -e "\t\tSecurity group: default"
echo -e "\t\tKey name: $KEYNAME"
```

```
openstack server create --flavor $FLAVOUR --image $IMAGE --nic net-id=$NIC
--security-group default --key-name $KEYNAME $INSTANCE
```

```
openstack server create --flavor $FLAVOUR --image $IMAGE --nic net-id=$NIC
--security-group default --key-name $KEYNAME $INSTANCE
```

```
while [ "$\{(openstack server list | grep $INSTANCE | awk '{print $6}')\}" != 'ACTIVE' ]; do
  printf ". 
  sleep 2
done
```

```
echo; echo "Creating volume $VOLNAME"
openstack volume create --size 1 $VOLNAME
```

```
echo; echo "Adding volume $VOLNAME to VM instance $INSTANCE"
openstack server add volume $INSTANCE $VOLNAME
```

```
openstack volume list
```

```
echo; echo
# END
EOM
```

osbash@controller:~$ chmod +x ~/instance_launch.sh
18.5 Appendix 5 - Script to launch a network with VMs

This script if ran on the controller node after the OpenStack Labs install will create four VM instances with two connected to a private network and two connected to the provider network with a router between them. Make sure to run `$OS_LAB/nat_tables.sh` on the hypervisor host to enable routing from the VM instances.

```bash
osbash@controller:~$ cat <<'EOM' > ~/network_launch.sh
#!/bin/bash

#######################
# network_launch.sh    #
# Diarmuid O'Briain    #
#######################

# Run this script on the controller node
# Access the Controller node
# KVM/QEMU:   ssh osbash@10.0.0.11
# VirtualBox: ssh -p 2230 osbash@localhost

# Make sure to run "sudo $OS_LAB/nat_tables.sh" on hypervisor host
# Create static route to SUBNET on hypervisor host

# Variables
INSTANCE_A=( host1 host2 )
INSTANCE_B=( host3 host4 )
FLAVOUR='m1.nano'
IMAGE='cirros'
PNET='PRIV-NET'
PSUBNET='PRIV-SUBNET'
DNS='8.8.8.8'
SUBNET='192.168.95.0/24'
SUBNET_UPPR='192.168.95'
PROVIDER_NIC=''
PNET_NIC=''
ROUTER='router1'

## Function ##
function host_create() {
    local _INSTANCE=$1
    local _FLAVOUR=$2
    local _IMAGE=$3
    local _NIC=$4
    echo; echo "Creating and launching instance $_INSTANCE with:"
    echo -e "\n\tFlavour: $_FLAVOUR"
    echo -e "\tImage: $_IMAGE"
    echo -e "\tNetwork UUID=$_NIC"
    echo -e "\tSecurity group: default"
    openstack server create --flavor $_FLAVOUR --image $_IMAGE --nic net-id=$_NIC
    --security-group default $_INSTANCE
}

## END FUNCTION ##

echo; echo "Setting admin-openrc variables"

export OS_PROJECT_DOMAIN_NAME=Default
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=admin_user_secret
export OS_AUTH_URL=http://controller:35357/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2

echo; echo "Creating flavour $FLAVOUR"
```

---

190  OpenStack Training Laboratory

---
openstack flavor create --id 0 --vcpus 1 --ram 1024 --disk 1 $FLAVOUR

echo; echo "Setting demo-openrc variables"
export OS_PROJECT_NAME=demo
export OS_USERNAME=demo
export OS_PASSWORD=demo_user_pass
export OS_AUTH_URL=http://controller:5000/v3

echo; echo "Adding port 22 (SSH) and ICMP to default security group"
openstack security group rule create --proto tcp --dst-port 22 default
openstack security group rule create --proto icmp default

echo; echo "Creating private network $PNET"
openstack network create $PNET
openstack subnet create --network $PNET --subnet-range $SUBNET --gateway $SUBNET_UPPR.1 --ip-version 4 --allocation-pool start=$SUBNET_UPPR.10,end=$SUBNET_UPPR.20 --dns-nameserver $DNS $PSUBNET

echo;echo "Extracting provider and $PNET network UIDs"
PROVIDER_NIC=$(openstack network list | grep provider | awk '{print $2}');
PNET_NIC=$(openstack network list | grep $PNET | awk '{print $2}');

for i in ${INSTANCE_A[@]}; do
    host_create $i $FLAVOUR $IMAGE $PROVIDER_NIC
done

for i in ${INSTANCE_B[@]}; do
    host_create $i $FLAVOUR $IMAGE $PNET_NIC
done

echo; echo "Server list"
openstack server list

openstack router create $ROUTER
openstack router set --external-gateway=provider $ROUTER

openstack router add subnet $ROUTER $PSUBNET

openstack router show $ROUTER

echo; echo

# END
EOM

osbash@controller:~$ chmod +x ~/network_launch.sh
18.6 Appendix 6 - stacktrain cluster creation script – KVM

```
ada:~$ cd $OS_ST
ada:~/OpenStack-lab/labs$ ./st.py --build cluster --provider kvm
INFO Using provider kvm.
INFO stacktrain start at Sat Sep 23 22:31:11 2017
INFO Asked to delete VM base.
INFO not found
WARNING There is no file at given path: /home/alovelace/OpenStack-lab/labs/img/ubuntu-16.04.3-server-amd64.iso
INFO Downloading http://releases.ubuntu.com/16.04/ubuntu-16.04.3-server-amd64.iso to /home/alovelace/OpenStack-lab/labs/img/ubuntu-16.04.3-server-amd64.iso
INFO This may take a while.
INFO Download succeeded.
INFO Install ISO:
/home/alovelace/OpenStack-lab/labs/img/ubuntu-16.04.3-server-amd64.iso
INFO base_fixups.sh -> 00_base_fixups.sh
INFO apt_init.sh -> 01_apt_init.sh
INFO apt_upgrade.sh -> 02_apt_upgrade.sh
INFO pre_download.sh -> 03_pre_download.sh
INFO apt_pre-download.sh -> 04_apt_pre-download.sh
INFO enable_osbash_ssh_keys.sh -> 05_enable_osbash_ssh_keys.sh
INFO zero_empty.sh -> 06_zero_empty.sh
INFO shutdown.sh -> 07_shutdown.sh
[sudo] password for alovelace: babbage
WARNING Graphics requested but DISPLAY is not set. Not running virt-viewer.
```

Starting install...
Creating domain... 0 B 00:00:00
Domain installation still in progress. Waiting for installation to complete.
INFO Waiting 5 seconds for VM base to come up.
INFO Booting into distribution installer.
INFO Initiating boot sequence for base.
INFO Waiting for VM base to be defined.
INFO Waiting for MAC address.
INFO Waiting for IP address.
INFO Waiting for ping returning from 192.168.122.47.
INFO Waiting for ssh server in VM base to respond at 192.168.122.47:22.
WARNING Adjusting permissions for key file (0400): /home/alovelace/OpenStack-lab/labs/lib/osbash-ssh-keys/osbash_key
Opening the ssh port on VM base.
```
Domain has shutdown. Continuing.
Domain creation completed.
Restarting guest.
```
INFO Connected to ssh server.
INFO Start autostart/00_base_fixups.sh
INFO done
INFO Start autostart/01_apt_init.sh
INFO done
INFO Start autostart/02_apt_upgrade.sh
INFO done
INFO Start autostart/03_pre-download.sh
INFO done
INFO Start autostart/04_apt_pre-download.sh
INFO done
INFO Start autostart/05_enable_osbash_ssh_keys.sh
INFO done
INFO Start autostart/06_zero_empty.sh
INFO done
INFO Start autostart/07_shutdown.sh
```

OpenStack Pike

02 Oct 2017
INFO done
INFO Start autostart/07_shutdown.sh
INFO done
INFO Processing of scripts successful.
INFO Waiting for shutdown of VM base.
[sudo] password for alovelace: babbage
INFO Compacting base-ssh-pike-ubuntu-16.04-amd64.
WARNING No virt-sparsify executable found.
WARNING Consider installing libguestfs-tools.
INFO Base disk created.
INFO stacktrain base disk build ends.
INFO Basedisk build took 8489 seconds
INFO Creating mgmt network: 10.0.0.0.
INFO Creating provider network: 203.0.113.0.
INFO Asked to delete VM controller.
INFO not found
INFO Creating copy-on-write VM disk.
WARNING Graphics requested but DISPLAY is not set. Not running virt-viewer.
WARNING No console to launch for the guest, defaulting to --wait -1

Starting install...
Creating domain... | 0 B 00:00:00
Domain creation completed.
You can restart your domain by running:
   virsh --connect qemu:///system start controller
INFO Waiting for VM controller to be defined.
INFO Node controller created.
INFO init_xxx_node.sh -> 00_init_controller_node.sh
INFO etc_hosts.sh -> 01/etc_hosts.sh
INFO enable_osbash_ssh_keys.sh -> 02_enable_osbash_ssh_keys.sh
INFO copy_openrc.sh -> 03_copy_openrc.sh
INFO apt_install_mysql.sh -> 04_apt_install_mysql.sh
INFO install_rabbitmq.sh -> 05_install_rabbitmq.sh
INFO install_memcached.sh -> 06_install_memcached.sh
INFO setup_keystone.sh -> 07_setup_keystone.sh
INFO get_auth_token.sh -> 08_get_auth_token.sh
INFO setup_glance.sh -> 09_setup_glance.sh
INFO setup_nova_controller.sh -> 10_setup_nova_controller.sh
INFO setup_neutron_controller.sh -> 11_setup_neutron_controller.sh
INFO setup_self-service_controller.sh -> 12_setup_self-service_controller.sh
INFO setup_neutron_controller_part_2.sh -> 13_setup_neutron_controller_part_2.sh
INFO setup_horizon.sh -> 14_setup_horizon.sh
INFO setup_cinder_controller.sh -> 15_setup_cinder_controller.sh
INFO setup_heat_controller.sh -> 16_setup_heat_controller.sh
INFO Starting VM controller
INFO Waiting for VM controller to run.
INFO Waiting for MAC address.
INFO Waiting for IP address.

.........
INFO Waiting for ssh server in VM controller to respond at 192.168.122.47:22.
INFO Connected to ssh server.
INFO Start autostart/00_init_controller_node.sh
INFO done
INFO Start autostart/01/etc_hosts.sh
INFO done
INFO Start autostart/02_enable_osbash_ssh_keys.sh
INFO done
INFO Start autostart/03_copy_openrc.sh
INFO done
INFO Start autostart/04_apt_install_mysql.sh
INFO done
INFO Start autostart/05_install_rabbitmq.sh
INFO done
INFO Start autostart/06_install_memcached.sh
INFO done
INFO Start autostart/07_setup_keystone.sh
..........................................................................................
INFO done
INFO Start autostart/08_get_auth_token.sh
..........................................................................
INFO done
INFO Start autostart/09_setup_glance.sh
..........................................................................................
INFO done
INFO Start autostart/10_setup_nova_controller.sh
..........................................................................................
INFO done
INFO Start autostart/11_setup_neutron_controller.sh
..........................................................
INFO done
INFO Start autostart/12_setup_self-service_controller.sh
..........................................................
INFO done
INFO Start autostart/13_setup_neutron_controller_part_2.sh
..........................................................................
INFO done
INFO Start autostart/14_setup_horizon.sh
..........................................................................
INFO done
INFO Start autostart/15_setup_cinder_controller.sh
..........................................................................
INFO done
INFO Start autostart/16_setup_heat_controller.sh
..........................................................................................
INFO done
INFO Processing of scripts successful.
INFO Asked to delete VM compute1.
[sudo] password for alovelace: babbage
INFO not found
INFO Creating copy-on-write VM disk.
INFO Adding empty disk to compute1: compute1-sdb
WARNING Graphics requested but DISPLAY is not set. Not running virt-viewer.
WARNING No console to launch for the guest, defaulting to --wait -1

Starting install...
Creating domain... | 0 B 00:00:00
Domain creation completed.
You can restart your domain by running:
   virsh --connect qemu:///system start compute1
INFO Waiting for VM compute1 to be defined.
INFO Node compute1 created.
INFO init_xxx_node.sh -> 00_init_computel_node.sh
INFO etc_hosts.sh -> 01/etc_hosts.sh
INFO enable_osbash_ssh_keys.sh -> 02_enable_osbash_ssh_keys.sh
INFO copy_openrc.sh -> 03_copy_openrc.sh
INFO setup_nova_compute.sh -> 04_setup_nova_compute.sh
INFO setup_neutron_compute.sh -> 05_setup_neutron_compute.sh
INFO setup_self-service_compute.sh -> 06_setup_self-service_compute.sh
INFO setup_neutron_compute_part_2.sh -> 07_setup_neutron_compute_part_2.sh
INFO setup_cinder_volumes.sh -> 08_setup_cinder_volumes.sh
INFO Starting VM compute1
INFO Waiting for VM compute1 to run.
INFO Waiting for MAC address.
INFO Waiting for IP address.

INFO Waiting for ssh server in VM compute1 to respond at 192.168.122.64:22.
INFO Connected to ssh server.
INFO Start autostart/00_init_computel_node.sh
INFO done
INFO Start autostart/01/etc_hosts.sh
INFO done

...
INFO  Start autostart/02_enable_osbash_ssh_keys.sh
INFO  done
INFO  Start autostart/03_copy_openrc.sh
INFO  done
INFO  Start autostart/04_setup_nova_compute.sh
INFO  done
INFO  Start autostart/05_setup_neutron_compute.sh
INFO  done
INFO  Start autostart/06_setup_self-service_compute.sh
INFO  done
INFO  Start autostart/07_setup_neutron_compute_part_2.sh
INFO  done
INFO  Start autostart/08_setup_cinder_volumes.sh
INFO  done
INFO  Processing of scripts successful.
INFO  Shutting down VM controller.
INFO  Waiting for shutdown of VM controller.
INFO  config_public_network.sh -> 00_config_public_network.sh
INFO  config_private_network.sh -> 01_config_private_network.sh
INFO  Starting VM controller
INFO  Waiting for VM controller to run.
INFO  Waiting for ssh server in VM controller to respond at 192.168.122.47:22.
INFO  Connected to ssh server.
INFO  Start autostart/00_config_public_network.sh
INFO  done
INFO  Start autostart/01_config_private_network.sh
INFO  done
INFO  Processing of scripts successful.
INFO  Shutting down VM compute1.
INFO  Waiting for shutdown of VM compute1.
INFO  Starting VM controller
INFO  Waiting for VM controller to run.
INFO  Waiting for ssh server in VM controller to respond at 192.168.122.47:22.
INFO  Connected to ssh server.
INFO  Processing of scripts successful.
INFO  Starting VM compute1
INFO  Waiting for VM compute1 to run.
INFO  Waiting for ssh server in VM compute1 to respond at 192.168.122.64:22.
INFO  Connected to ssh server.
INFO  Processing of scripts successful.
INFO  Cluster build took 2006 seconds
Your cluster nodes:
INFO  VM name: compute1
INFO  SSH login: ssh osbash@192.168.122.64
INFO  (password: osbash)
INFO  VM name: controller
INFO  SSH login: ssh osbash@192.168.122.47
INFO  (password: osbash)
INFO  Dashboard: Assuming horizon is on controller VM.
INFO  http://192.168.122.47/horizon/
INFO  User : demo (password: demo_user_pass)
INFO  User : admin (password: admin_user_secret)
INFO  Network: mgmt
INFO  Network address: 10.0.0.0
INFO  Network: provider
INFO  Network address: 203.0.113.0
18.7 Appendix 7 - stacktrain cluster creation script – VirtualBox

ada:~$ cd $OS_ST
ada:~/Openstack-lab/labs $ st.py --build cluster
INFO Using provider virtualbox.
INFO stacktrain start at Fri Sep 22 16:24:57 2017
INFO Creating
/home/alovelace/OpenStack-lab/labs/img/base-ssh-pike-ubuntu-16.04-amd64.vdi.
INFO ISO image okay.
INFO Install ISO:
/home/alovelace/OpenStack-lab/labs/img/ubuntu-16.04.3-server-amd64.iso
INFO Asked to delete VM base
INFO not found
INFO Created VM base.
INFO Attaching to VM base:
/home/alovelace/OpenStack-lab/labs/img/ubuntu-16.04.3-server-amd64.iso
INFO Creating disk (size: 10000 MB):
/home/alovelace/OpenStack-lab/labs/img/tmp-disk.vdi
INFO Attaching to VM base:
/home/alovelace/OpenStack-lab/labs/img/tmp-disk.vdi
INFO base_fixups.sh -> 00_base_fixups.sh
INFO apt_init.sh -> 01_apt_init.sh
INFO apt_upgrade.sh -> 02_apt_upgrade.sh
INFO pre-download.sh -> 03_pre-download.sh
INFO apt_pre-download.sh -> 04_apt_pre-download.sh
INFO enable_osbash_ssh_keys.sh -> 05_enable_osbash_ssh_keys.sh
INFO zero_empty.sh -> 06_zero_empty.sh
INFO shutdown.sh -> 07_shutdown.sh
INFO Booting VM base.
INFO Starting VM base with headless GUI
INFO Waiting 10 seconds for VM base to come up.
INFO Booting into distribution installer.
INFO Initiating boot sequence for base.
INFO Waiting for ssh server in VM base to respond at 127.0.0.1:2229.
WARNING Adjusting permissions for key file (0400):
/home/alovelace/OpenStack-lab/labs/lib/osbash-ssh-keys/osbash_key
..............................................................................
..............................................................................
..............................................................................
....................................................
INFO Connected to ssh server.
INFO Start autostart/00_base_fixups.sh
INFO done
INFO Start autostart/01_apt_init.sh
..............................................................................
....................................................
INFO done
INFO Start autostart/02_apt_upgrade.sh
..............................................................................
....................................................
INFO done
INFO Start autostart/03_pre-download.sh
..............................................................................
....................................................
INFO done
INFO Start autostart/04_apt_pre-download.sh
..............................................................................
..............................................................................
..............................................................................
..............................................................................
..............................................................................
....................................................
INFO done
INFO Start autostart/05_enable_osbash_ssh_keys.sh
INFO done
INFO Start autostart/06_zero_empty.sh
........
INFO done
INFO Start autostart/07_shutdown.sh
...
INFO done
Processing of scripts successful.
Waiting for shutdown of VM base.

Machine powered off.
Detaching disk from VM base.
Unregistering and deleting VM: base
Compacting /home/alovelace/OpenStack-lab/labs/img/tmp-disk.vdi.
Unregistering disk /home/alovelace/OpenStack-lab/labs/img/tmp-disk.vdi
Base disk created.
Moving base disk to: /home/alovelace/OpenStack-lab/labs/img/base-ssh-pike-ubuntu-16.04-amd64.vdi
Base disk build ended.
Basedisk build took 4622 seconds
Creating mgmt network: 10.0.0.0.
Creating host-only interface.
Configuring host-only network mgmt with gw address 10.0.0.1 (vboxnet4).
Creating provider network: 203.0.113.0.
Creating host-only interface.
Configuring host-only network provider with gw address 203.0.113.1 (vboxnet5).

 Asked to delete VM controller
not found
Created VM controller.
Attaching to VM controller (multi): /home/alovelace/OpenStack-lab/labs/img/base-ssh-pike-ubuntu-16.04-amd64.vdi
Node controller created.

init_xxx_node.sh -> 00_init_controller_node.sh
etc_hosts.sh -> 01/etc_hosts.sh
enable_osbash_ssh_keys.sh -> 02_enable_osbash_ssh_keys.sh
copy_openrc.sh -> 03_copy_openrc.sh
apt_install_mysql.sh -> 04_apt_install_mysql.sh
install_rabbitmq.sh -> 05_install_rabbitmq.sh
install_memcached.sh -> 06_install_memcached.sh
setup_keystone.sh -> 07_setup_keystone.sh
get_auth_token.sh -> 08_get_auth_token.sh
setup_glance.sh -> 09_setup_glance.sh
setup_nova_controller.sh -> 10_setup_nova_controller.sh
setup_neutron_controller.sh -> 11_setup_neutron_controller.sh
setup_self-service_controller.sh -> 12_setup_self-service_controller.sh
setup_neutron_controller_part_2.sh -> 13_setup_neutron_controller_part_2.sh
setup_horizon.sh -> 14_setup_horizon.sh
setup_cinder_controller.sh -> 15_setup_cinder_controller.sh
setup_heat_controller.sh -> 16_setup_heat_controller.sh
Starting VM controller with headless GUI
Waiting for ssh server in VM controller to respond at 127.0.0.1:2230.

Connected to ssh server.
Start autostart/00_init_controller_node.sh
done
Start autostart/01/etc_hosts.sh
done
Start autostart/02_enable_osbash_ssh_keys.sh
done
Start autostart/03_copy_openrc.sh
done
Start autostart/04_apt_install_mysql.sh
done
Start autostart/05_install_rabbitmq.sh
done
Start autostart/06_install_memcached.sh
done
Start autostart/07_setup_keystone.sh

INFO done
INFO Start autostart/08_get_auth_token.sh
INFO done
INFO Start autostart/09_setup_glance.sh
.............................................
INFO done
INFO Start autostart/10_setup_nova_controller.sh
.............................................
INFO done
INFO Start autostart/11_setup_neutron_controller.sh
.............................................
INFO done
INFO Start autostart/12_setup_self-service_controller.sh
.............................................
INFO done
INFO Start autostart/13_setup_neutron_controller_part_2.sh
.............................................
INFO done
INFO Start autostart/14_setup_horizon.sh
.............................................
INFO done
INFO Start autostart/15_setup_cinder_controller.sh
.............................................
INFO done
INFO Start autostart/16_setup_heat_controller.sh
.............................................
INFO done
INFO Processing of scripts successful.
INFO Asked to delete VM compute1
INFO      not found
INFO Created VM compute1.
INFO Attaching to VM compute1 (multi):
/home/alovelace/OpenStack-lab/labs/img/base-ssh-pike-ubuntu-16.04-amd64.vdi
INFO Attaching disk (size: 204800 MB):
/home/alovelace/OpenStack-lab/labs/img/compute1-sdb.vdi
INFO Node compute1 created.
INFO init_xxx_node.sh -> 00_init_compute1_node.sh
INFO etc_hosts.sh -> 01_etc_hosts.sh
INFO enable_osbash_ssh_keys.sh -> 02_enable_osbash_ssh_keys.sh
INFO copy_openrc.sh -> 03_copy_openrc.sh
INFO setup_nova_compute.sh -> 04_setup_nova_compute.sh
INFO setup_neutron_compute.sh -> 05_setup_neutron_compute.sh
INFO setup_self-service_compute.sh -> 06_setup_self-service_compute.sh
INFO setup_neutron_compute_part_2.sh -> 07_setup_neutron_compute_part_2.sh
INFO setup_cinder_volumes.sh -> 08_setup_cinder_volumes.sh
INFO Starting VM compute1 with headless GUI
INFO Waiting for ssh server in VM compute1 to respond at 127.0.0.1:2232.
...........
INFO Connected to ssh server.
...
INFO Start autostart/00_init_compute1_node.sh
......
INFO done
INFO Start autostart/01_etc_hosts.sh
...
INFO done
INFO Start autostart/02_enable_osbash_ssh_keys.sh
INFO done
INFO Start autostart/03_copy_openrc.sh
INFO done
INFO Start autostart/04_setup_nova_compute.sh
.............................................
INFO done
INFO Start autostart/05_setup_neutron_compute.sh
.............................................
INFO done
INFO Start autostart/06_setup_self-service_compute.sh.
INFO done
INFO Start autostart/07_setup_neutron_compute_part_2.sh
.......
INFO done
INFO Start autostart/08_setup_cinder_volumes.sh
..........................INFO done
INFO Processing of scripts successful.
INFO Shutting down VM controller.
INFO Waiting for shutdown of VM controller.
...........
INFO Machine powered off.
INFO config_public_network.sh -> 00_config_public_network.sh
INFO config_private_network.sh -> 01_config_private_network.sh
INFO Starting VM controller with headless GUI
INFO Waiting for ssh server in VM controller to respond at 127.0.0.1:2230.
...........
INFO Connected to ssh server.
INFO Start autostart/00_config_public_network.sh
..........................INFO done
INFO done
INFO Start autostart/01_config_private_network.sh
..........................INFO done
INFO Processing of scripts successful.
INFO Shutting down VM controller.
INFO Waiting for shutdown of VM controller.
...........
INFO Machine powered off.
INFO Shutting down VM compute1.
INFO Waiting for shutdown of VM compute1.
.......
INFO Machine powered off.
INFO Starting VM controller with headless GUI
INFO Waiting for ssh server in VM controller to respond at 127.0.0.1:2230.
...........
INFO Connected to ssh server.
INFO Processing of scripts successful.
INFO Starting VM compute1 with headless GUI
INFO Waiting for ssh server in VM compute1 to respond at 127.0.0.1:2232.
...........
INFO Connected to ssh server.
INFO Processing of scripts successful.
INFO Cluster build took 1037 seconds
Your cluster nodes:
INFO VM name: compute1
INFO SSH login: ssh -p 2232 osbash@127.0.0.1
INFO (password: osbash)
INFO VM name: controller
INFO SSH login: ssh -p 2230 osbash@127.0.0.1
INFO (password: osbash)
INFO Dashboard: Assuming horizon is on controller VM.
INFO http://127.0.0.1:8880/horizon/
INFO User : demo (password: demo_user_pass)
INFO User : admin (password: admin_user_secret)
INFO Network: mgmt
INFO Network address: 10.0.0.0
INFO Network: provider
INFO Network address: 203.0.113.0
This page is intentionally blank
19. Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES</td>
<td>Advanced Encryption Standard</td>
</tr>
<tr>
<td>BIOS</td>
<td>Basic Input/Output System</td>
</tr>
<tr>
<td>Ceilometer</td>
<td>Telemetry service</td>
</tr>
<tr>
<td>Cinder</td>
<td>Block storage service</td>
</tr>
<tr>
<td>AMQP</td>
<td>Advanced Message Queuing Protocol</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>CRUD</td>
<td>Create, read, update and delete</td>
</tr>
<tr>
<td>CT</td>
<td>Container</td>
</tr>
<tr>
<td>DHCP</td>
<td>Dynamic Host Configuration Protocol</td>
</tr>
<tr>
<td>EC2</td>
<td>Elastic Compute 2 (Amazon basic VM)</td>
</tr>
<tr>
<td>XML</td>
<td>eXtensible Markup Language</td>
</tr>
<tr>
<td>GB</td>
<td>Gigabytes</td>
</tr>
<tr>
<td>Glance</td>
<td>Image service</td>
</tr>
<tr>
<td>HA</td>
<td>High Availability</td>
</tr>
<tr>
<td>Heat</td>
<td>Orchestration service</td>
</tr>
<tr>
<td>Horizon</td>
<td>Dashboard</td>
</tr>
<tr>
<td>HOT</td>
<td>Heat Orchestration Template</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>HVM</td>
<td>Hardware-assisted Virtual Machine</td>
</tr>
<tr>
<td>IaaS</td>
<td>Infrastructure as a Service</td>
</tr>
<tr>
<td>ICMP</td>
<td>Internet Control Message Protocol</td>
</tr>
<tr>
<td>I/O</td>
<td>Input/Output</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>Keystone</td>
<td>Identity service</td>
</tr>
<tr>
<td>KVM</td>
<td>Kernel Virtual Machine</td>
</tr>
<tr>
<td>L2</td>
<td>Layer-2 bridging/switching.</td>
</tr>
<tr>
<td>L3</td>
<td>Layer 3 - Routing</td>
</tr>
<tr>
<td>libvirt</td>
<td>Toolkit to manage virtualisation hosts</td>
</tr>
<tr>
<td>LM</td>
<td>Long Mode</td>
</tr>
<tr>
<td>MB</td>
<td>Megabytes</td>
</tr>
<tr>
<td>LVM</td>
<td>Logical Volume Manager</td>
</tr>
<tr>
<td>NASA</td>
<td>National Aeronautics and Space Administration</td>
</tr>
<tr>
<td>NAT</td>
<td>Masquerading - Network Address Translation</td>
</tr>
<tr>
<td>Neutron</td>
<td>Networking service</td>
</tr>
<tr>
<td>Nova</td>
<td>Compute service</td>
</tr>
<tr>
<td>NTP</td>
<td>Network Time Protocol</td>
</tr>
<tr>
<td>ORM</td>
<td>Object Relational Mapper</td>
</tr>
<tr>
<td>OvS</td>
<td>Open vSwitch</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Full Form</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
</tr>
<tr>
<td>PaaS</td>
<td>Platform as a Service</td>
</tr>
<tr>
<td>PNI</td>
<td>Physical Networking Infrastructure</td>
</tr>
<tr>
<td>QCOW2</td>
<td>QEMU Copy On Write</td>
</tr>
<tr>
<td>QEMU</td>
<td>Quick Emulator</td>
</tr>
<tr>
<td>RabbitMQ</td>
<td>Rabbit Message Queue</td>
</tr>
<tr>
<td>RADOS</td>
<td>Reliable Autonomic Distributed Object Store</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>RPC</td>
<td>Remote Procedure Call</td>
</tr>
<tr>
<td>SaaS</td>
<td>Software as a Service</td>
</tr>
<tr>
<td>SDN</td>
<td>Software Defined Networking</td>
</tr>
<tr>
<td>SPICE</td>
<td>Simple Protocol for Independent Computing Environments</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
<tr>
<td>SSH</td>
<td>Secure Shell</td>
</tr>
<tr>
<td>STONITH</td>
<td>Shoot The Offending Node In The Head</td>
</tr>
<tr>
<td>SVM</td>
<td>Secure Virtual Machine</td>
</tr>
<tr>
<td>Swift</td>
<td>Object storage service</td>
</tr>
<tr>
<td>Trove</td>
<td>Database service</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>UUID</td>
<td>Universally Unique IDentifier</td>
</tr>
<tr>
<td>vCPU</td>
<td>virtual Central Processing Unit</td>
</tr>
<tr>
<td>virsh</td>
<td><em>libvirt</em> based command line interface tool for managing guests and the hypervisor.</td>
</tr>
<tr>
<td>VM</td>
<td>Virtual Machine</td>
</tr>
<tr>
<td>VNC</td>
<td>Virtual Network Computing</td>
</tr>
<tr>
<td>VMX</td>
<td>Virtual Machine eXtensions</td>
</tr>
<tr>
<td>VNI</td>
<td>Virtual Networking Infrastructure</td>
</tr>
<tr>
<td>VT-x</td>
<td>Virtualisation Technology - x86 architectures</td>
</tr>
<tr>
<td>XCP</td>
<td>Xen Cloud Platform</td>
</tr>
</tbody>
</table>
20. Bibliography


This page is intentionally blank