Opennebula and The Xen Hypervisor

http://cloudpointers.wordpress.com

Marc Reilly       Michael O’ Cearra
12/20/2012

marcreilly2007@gmail.com
@marcreilly
michael290988@hotmail.com
## Table of Contents

Hybrid cloud design .................................................................................................................. 3

Operating systems .................................................................................................................... 3
Debian ..................................................................................................................................... 3
OpeneSuse ................................................................................................................................. 3
Ubuntu Server ........................................................................................................................... 3
Management tools ..................................................................................................................... 4
OpenNebula ............................................................................................................................... 4
XenServer ................................................................................................................................. 5
XCP .......................................................................................................................................... 7
Possible Designs ....................................................................................................................... 8
Example 1 ................................................................................................................................ 8
Example 2 ................................................................................................................................ 8
Chosen design ........................................................................................................................... 9

Private Cloud Installation Guide ............................................................................................. 10
Pre-requisites ............................................................................................................................ 10
Configuring the nodes ............................................................................................................... 13
Node prerequisites .................................................................................................................... 13
Node Configuration .................................................................................................................. 15
Launching a virtual machine ..................................................................................................... 32
Installing Sunstone self service portal ..................................................................................... 33

Provisioning of the public cloud ............................................................................................... 36

1. Register an account with amazon web services .................................................................. 36
2. Set up ssh keys on public cloud .......................................................................................... 36
3. Generate x.509 certificate to authenticate openenbula with AWS .................................. 36
4. Download the aws api ........................................................................................................ 36
5. Then we must configure Opennebula drivers and set the EC2 information manager configuration .................................................................................................................. 36
6. EC2 API Tools configuration ............................................................................................... 36
7. Add AWS host to Opennebula ............................................................................................. 37
8. Launch instance .................................................................................................................... 37

Conclusion ................................................................................................................................. 38
Hybrid cloud design

When deciding which was the best possible design for building our hybrid cloud, we came up with a wide range of cloud management tools, monitoring tools and Operating systems for which to install these components onto. We are going to explain from here, which different components we tried out, the features which they offered and the reasons for which we ended up choosing the hypervisor, management tools, monitoring tools and operating systems that we did. We will firstly talk about each section then lastly show and explain the design which we stuck too.

Operating systems

Debian

Debian is a free and open source operating system made up of a series of software packages. It includes GNU operating system tools and a Linux kernel. It has proven to be one of the most popular Linux distributions out there. It is compatible with Xen and has a GUI integrated with it.

Problem : The problem we had with this is that once we had it installed, while we were installing Xen onto it, we kept getting various errors along with the internet not working which at the time we felt better to try another operating system.

OpenSuse

OpenSuse is also a free and open source operating system made up of a series of software packages. It includes GNU operating system tools and a Linux kernel. It is compatible with Xen and includes both a default graphical user interface (GUI) and a command line interface option.

Problem : we also encountered problems along with installing Xen as when we were attempting to integrate it with Xen, we came across errors which were very time consuming to overcome so we felt the best course of action was to try another operating system.

Ubuntu Server

Ubuntu 10.04 Server Edition can also run on VMware ESX Server, Oracle's VirtualBox and VM, Citrix Systems XenServer hypervisors, Microsoft Hyper-V, as well as Kernel-based Virtual Machine. Its firewall is extended to common services used by the operating system. Ubuntu 10.04 LTS Server Edition supports two major architectures: Intel x86 and AMD64. The server edition uses a screen
mode character-based interface for the installation, instead of a graphical installation process. It consists of the open core Eucalyptus, libvirt, KVM or Xen virtualization technology.

Problems: we had none with this, it had no GUI so we worked on its command prompt but other than a few errors, we were able to install and configure Xen onto it successfully.

Management tools

OpenNebula

OpenNebula is an open sourced cloud computing management application which allows for it to manage various different structure of datacentres and clouds. It has the ability to manage private, public and hybrid clouds. It helps bind a range of technologies such as storage, network, virtualization, monitoring and security to help form a platform which deploys multi-tier services as virtual machines combining different cloud structures. Its features allows for integration, management, scalability, security and so on. It also emphasises that it can promise standardisation, interoperability and portability giving cloud users a wide range of cloud interfaces to choose from including EC2 Query and vCloud and hypervisors such as Xen. It can accommodate multiple hardware and software combinations in a data centre. Its features include:

- **User management**: It is possible to configure multiple users, who will have access only to their own instances, the ability to account for used resources, and with limits enforced by quota

- **VM Image management**: Every disk image is registered and managed by a centralized image catalog

- **Virtual Network management**: It is possible to define multiple networks bonded to different physical interfaces, with either static or dynamic IP address assignment

- **Virtual Machine management**: Every machine has its own set of characteristics (for example, CPU, memory, disk storage, and virtual network) and can be launched under every available hypervisor of our cluster

- **Service management**: A group of virtual machines can be grouped for being deployed together at boot time, and every virtual machine can be configured at boot time, without the need to assign different disk images for similar machines

- **Infrastructure management**: The physical hosts can be managed alone or grouped on independent clusters, and it is useful when you have a heterogeneous environment
• **Storage management**: The support for most common storage solutions is found in data centers such as shared storage such as **Network Attached Storage (NAS)** with specific support for optimal disk image management

• **Information management**: Every host and every virtual machine is actively monitored every few seconds, and it is already available in integration with standard monitoring tools such as Ganglia

• **Scheduling**: Virtual machines are deployed on host nodes following specific user requirements and resource-aware policies, such as packing, striping, or load-aware

• **User interface**: It includes the command-line tools available for managing every aspect of OpenNebula.

• **Operations center**: Most of the information and tasks available from the command line are available on web interfaces browsable with any modern web browser on any operating system

For a Hybrid cloud, which uses both local and remote resources, the two main features available are as follows:

• **Cloud-bursting**: It is the ability to add computing resources to your local infrastructure, using external resources, in order to meet peak demands or implement high-availability/disaster recovery strategies. This is essential for having a flexible and reliable infrastructure.

• **Federation**: It is the ability to combine together different clusters, dislocated in different physical positions, enabling higher levels of scalability and reliability.

Problems: We found no problems with it whatsoever, judging by the features which are described, we came up with the conclusion that it was the best suited management tool for our cloud.

**XenServer**

XenServer is an enterprise-ready, commercial virtualization platform that contains all the capabilities in which to create and manage a virtual infrastructure. Some of its features include:

**Datacenter automation with XenServer**: With Citrix XenServer, organizations can automate key IT processes to improve service delivery and business continuity for virtual environments resulting in both time and money savings while providing more responsive IT services.
Site Recovery: Provides site-to-site disaster recovery planning and services for virtual environments. Site recovery is easy to set up, fast to recover, and has the ability to frequently test to ensure disaster recovery plans remain valid.

Dynamic Workload Balancing: Improves system utilization and increases application performance by automatically balancing two virtual machines within a resource pool. Workload balancing intelligently places VMs on the most suitable host in the resource pool by matching application requirements to available hardware resources.

High Availability: Automatically restarts virtual machines if a failure occurs at the VM, hypervisor, or server level. The auto restart capability allows users to protect all virtualized applications and bring higher levels of availability to the business.

Automated VM Protection and Recovery: Utilizing an easy set-up wizard, administrators can create snapshot and archival policies. Regularly scheduled snapshots help to protect against data loss in case of a VM failure. The policies established are based on snapshot type, frequency, amount of historical data that is retained, and an archive location. Recovering a VM is completed by simply choosing the last good known archive.

Memory Optimization: Reduces costs and improves application performance and protection by sharing unused server memory between VMs on the host server.

Storage XenMotion: Move live running virtual machines and their associated virtual disk image within and across resource pools leveraging local and shared storage. This enables users to move a VM and its VDI from a development to production environment, move between tiers of storage when a VM is limited by storage capacity, and perform maintenance and upgrades with zero downtime. Conversion Tools

XenMotion: Citrix XenMotion eliminates the need for planned downtime by enabling active virtual machines to be moved to a new host with no application outages or downtime

Web Console with Delegated Admin: Web Self Service provides IT administrators with a simple web-based console to delegate individual VM rights to application owners as well as a way for application owners to manage day to day operations of their VM's.

Provisioning Services: Reduce storage requirements by creating a set of golden images which can be streamed to both physical and virtual servers for fast, consistent, and reliable application deployments.

Distributed Virtual Switching: Create a multi-tenant, highly secure and extremely flexible network fabric that allows VMs to move freely within the network while maintaining security and control.

XenServer Conversion Manager: Automate the process of converting VMware virtual machines into XenServer virtual machines with this simple batch conversion tool.

Heterogeneous Pools: Enables resource pools to contain servers with different processor types, and support full XenMotion, high availability, workload balancing, and shared storage functionality.
Problem: The main problem we had with this virtualisation management was that we had to use open source software and we then came aware that this was commercial which then made it impossible for us to use it.

XCP

Xen Cloud Platform is an open source management virtualization provided for cloud computing. It includes the Xen hypervisor, an enterprise ready XEN API toolstack and the ability to integrate for cloud, storage and networking solutions. Some other features include:

- VM lifecycle: live snapshots, checkpoint, migration
- Resource pools: flexible storage and networking
- Event tracking: progress, notification
- Upgrade and patching capabilities
- Real-time performance monitoring and alerting
- Built-in support and templates for Windows and Linux guests
- Open vSwitch support built-in
- Storage XenMotion® live Migration (cross-pool migration, VDI migration)

Problem: The problem which we had with that was the fact there were some many errors when integrating it with Xen which made us believe that there was a better virtualisation management which would be more suited to Xen.

Final outcome: We decided to work with UbuntuServer as the operating system because as stated before, it was working perfectly with Xen with also the added security of just using the command line instead of a GUI appealed to us. We decided to us OpenNebula for a various amount of reasons, it had a monitoring tool built into it which appealed to us as integration between OpenNebula and the tool would be no issue, we also liked the look of it as it took note of all the important aspects of the cloud. It also had load balancing integrated with it which would be a big help if we were to attempt to push it into the public cloud and lastly it gave us more information on the VM’s we instantiated rather then just using Xen itself.
Possible Designs

Example 1

For the first possible design, we had two hosts, one would have Xen on it, the second host would have OpenNebula on it and both would have storage operated by SSH. Whenever we needed to deploy an image we would send it over the two hosts, this means a lot of time will be taken up transferring these images. What is happening here is firstly the network, image and template are created in OpenNebula, and when the VM is instantiated, everything is sent over to the other host. The main problem here is the waiting time when the image was being transferred as it was taking far too long. The other problem here was the storage was a point of failure, it wasn’t shared but was also in the same host as the virtualisation management and the hypervisor respectively.

Example 2

For the first possible design, we had two hosts, one would have Xen on it, the second host would have OpenNebula on it and both would have storage operated by shared storage. This would mean that deploying an image from one host to another is unnecessary as both hosts are sharing the storage. What is happening here is firstly the network, image and template are created in OpenNebula, and when the VM is instantiated, everything is sent over to the other host. This architecture is different to the first design as this has shared storage meaning there was no waiting time waiting for the image to transfer. The bad point here was the same with the first design.
thought in that the storage was a one point failure as if one of the storages went down, then either the hypervisor or virtualisation management would be affected also.

This was the third and final design which we picked. What is happening here is firstly the network, image and template are created in OpenNebula, and when the VM is instantiated, everything is sent over to the other host. What is different is that /var/lib/one will be made available between replication server 1 and server2. Same folder will be exposed out using NFS share so that OpenNebula front end and OpenNebula Node together can share it. There will be high availability as there are two storage areas. There is heartbeat also present at all times. Shared storage is present, therefore all nodes have access to any update data in the storage. Both
storages are replicating off each other. Once a Vm has reached its max memory, it will then be pushed into the public cloud to EC2.

Private Cloud Installation Guide

Pre-requisites

Install and configure four virtual machines in vmware workstation. For this example I downloaded the ISO files and used these four Ubuntu virtual machines:

<table>
<thead>
<tr>
<th>Host Name</th>
<th>Operating System</th>
<th>Use</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server 1</td>
<td>Ubuntu server 12.10 LTS</td>
<td>Primary Network attached storage node</td>
<td>RAM: 1GB CPU: 1 PROCESSORS</td>
</tr>
<tr>
<td>Server 2</td>
<td>Ubuntu server 12.10 LTS</td>
<td>Secondary Network attached storage node</td>
<td>RAM: 1GB CPU: 1 PROCESSORS</td>
</tr>
<tr>
<td>oneserver</td>
<td>Ubuntu Desktop 12.04</td>
<td>Opennebula frontend</td>
<td>RAM: 1.2GB CPU: 2 PROCESSORS</td>
</tr>
<tr>
<td>Onehost</td>
<td>Ubuntu server 12.10 LTS</td>
<td>Opennebula Node (Xen hypervisor node)</td>
<td>RAM: 5GB CPU: 4 PROCESSORS</td>
</tr>
</tbody>
</table>

To create these virtual machines (VMs) in workstation
1. Open workstation
2. Click File and then new virtual machine
3. Select custom and next
4. Select next again and choose “I will install the operating system later” and continue
5. Then select linux and Ubuntu 64bit
6. On the next screen you will be greeted by the following screen. Name the vms as the configuration above. (do the same for memory and cpu also)
7. The next screen contains the network configuration, where you should select NAT. This will allow you to freely assign network configuration to your host VMs.

8. After this choose the defaults until you reach the “specify disk capacity” page. The configuration I’ve chosen is below but you are free to provision resources according to your needs.

<table>
<thead>
<tr>
<th>Host Name</th>
<th>Capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server 1</td>
<td>50 GB</td>
</tr>
<tr>
<td>Server 2</td>
<td>50 GB</td>
</tr>
<tr>
<td>Oneserver</td>
<td>60 GB</td>
</tr>
<tr>
<td>Onehost</td>
<td>100</td>
</tr>
</tbody>
</table>

9. Continue until you reach this screen. Select “customize hardware”.

10. Then select “new cd/dvd” select your ISO image and the connect on power on button. Click close and finish.
11. We will now configure the host network.

**VMWare Workstation**

**Workstation network configuration**
The next step is to configure the network in which the workstation vms will connect to:
1. Select edit and virtual network editor at the top of the window in workstation.
2. Select NAT and change the subnet IP Address to 192.168.1.0 and unselect DCHP.
3. Then select NAT settings and make sure the gateway IP is set to 192.168.1.1
4. Select ok and then then ok again
5. The host network is now configured for your vms and Your vms are now ready to be installed and configured

**Server 1 and server 2 installation**

Server 1 and server 2 are being used as network attached storage (NAS) replication servers. Therefore installation of both these servers will be identical apart from hostnames and IP addresses. To manually input network configuration press enter when the os is looking to find the network configuration using DCHP. Also when addition packages are offered for installation, select openssh.

The configuration used for these servers is

<table>
<thead>
<tr>
<th>Host Name</th>
<th>Username</th>
<th>Operating System</th>
<th>Network configuration</th>
<th>Partitioning Name</th>
<th>Size</th>
<th>Mount point</th>
</tr>
</thead>
<tbody>
<tr>
<td>server 1</td>
<td>localadmin</td>
<td>Ubuntu server 12.10 LTS</td>
<td>IP address: 192.168.1.94, Domain: xencloud.com</td>
<td>Sda1, Sda5</td>
<td>100 MB, 5GB</td>
<td>Boot, Root</td>
</tr>
<tr>
<td>server 2</td>
<td>localadmin</td>
<td>Ubuntu server 12.10 LTS</td>
<td>IP address: 192.168.1.95, Domain: xencloud.com</td>
<td>Sda6, Sda7, Sda8</td>
<td>1GB, 200MB, 60GB</td>
<td>Swap, Don’t mount, Don’t mount</td>
</tr>
</tbody>
</table>

When both these servers are installed using the recommended configurations we will move on to the oneserver setup.

**Oneserver Installation**
The oneserver node will be the Opennebula frontend. The front end is used to administer the hosts i.e. hypervisors. It is on the frontend that we will install Opennebula. The network configuration for the frontend is different and I shall go through it later on. When prompted insert/implement the following attributes.
### Onehost Installation

The onehost node is where we will install the hypervisor. The installation follows the same procedures as the NAS server except you must use the following configuration.

<table>
<thead>
<tr>
<th>Host Name</th>
<th>Username</th>
<th>Operating System</th>
<th>Network configuration</th>
<th>Partitioning</th>
<th>Size</th>
<th>Mount point</th>
</tr>
</thead>
<tbody>
<tr>
<td>onehost</td>
<td>localadmin</td>
<td>Ubuntu Desktop 12.04</td>
<td>IP address: 192.168.1.95 Domain: xencloud.com</td>
<td>Sda1 Sda5</td>
<td>55GB 5GB</td>
<td>Root swap</td>
</tr>
<tr>
<td>Onehost</td>
<td>localadmin</td>
<td>Ubuntu Server 12.10</td>
<td>IP address: 192.168.1.96 Domain: xencloud.com</td>
<td>Sda1 Sda5</td>
<td>95GB 5GB</td>
<td>Root swap</td>
</tr>
</tbody>
</table>

### Configuring the nodes

For the sake of transparency and ease of access we configured the nodes through a secure shell (ssh) via the onehost node. In order to do this we must first configure the network settings for onehost. To do this we must follow these steps

#### Node prerequisites

**VT-x enabled**

You must enable virtualization in the bios of the host machine and in workstation by accessing the vm’s settings and enabling vt-x.

![Virtual Machine Settings](image)

You must also go to the onehost.vmx file usually stored under “\Documents\Virtual Machines\xen onehost\xen onehost.vmx”. open that file in notepad and insert the following line, save it and HVM will be enabled.

```
hypervisor.cpuid.v0="FALSE"
```
**Oneserver network configuration**

- Boot the oneserver and login using localadmin
- Open a terminal and edit the network interfaces file. To do this we issue the following command

  ```
  Sudo pico /etc/network/interfaces
  ```

- We then edit the file so it looks like this

  ```
  auto lo
  iface lo inet loopback
  auto eth0
  iface eth0 inet static
  address 192.168.1.96
  netmask 255.255.255.0
  network 192.168.1.0
  broadcast 192.168.1.255
  gateway 192.168.1.1
  dns-nameservers 192.168.1.1
  dns-search xencloud.com
  ```

- We then save the new configuration (Ctrl+o) and exit (Ctrl + x)
- We must then add our new hosts to “/etc/hosts” file. Please make sure the following hosts are added.
  ```
  sudo pico /etc/hosts
  192.168.1.94 server1.xencloud.com server1
  192.168.1.95 server2.xencloud.com server2
  192.168.1.96 oneserver.xencloud.com oneserver
  192.168.1.97 onehost.xencloud.com onehost
  ```

- “Ctrl + o” then “ctrl + x”.
- sudo reboot

on reboot the vm should have network access. After installation you should update the packages using the command
sudo apt-get update

if the network was configured correctly apt should get updated.

**Secure shell setup**

Secure shell provides secure network services between nodes. We will use it to access remote nodes from the frontend. The first step is to make sure openssh-server is installed on all nodes. To install it run the following command on all nodes.

```
Sudo apt-get install openssh-server
```

You must then generate a keygen in each host by running the following command. It will assign a public and primary key to the node. You must not share your private key with anybody

```
ssh-keygen
```

After it is installed and the keys are generated in each node we can ssh to remote nodes using the following syntax

```
username@host e.g oneadmin@server1
```

The first time you connect it will ask you if you want to add the host to a list of known hosts, you must select yes. You will be then prompted for the remote host user’s password. After entering the password you should connect.(note the hostname on the terminal to make sure the connection was successful) You can also configure password-less access which we will go through later in the Opennebula installation.

**Node Configuration**

**Server 1 and server 2**

Server 1 and server 2 are replication servers providing high availability RAID1 in case of node failure. This is a very important feature in providing better service and un-interrupted service. The main packages used here are

- nfs-kernel-server
  - This package provides storage over the network
- drbd8
  - This is a package is used to provide highly available clusters by replication. When configured it will replicate the contents of the primary storage to the secondary storage. It can be called network based RAID1
- Heartbeat
  - This package allows nodes in a cluster to be aware of other nodes activity/inactivity. This is done by both nodes sending signals to each other and waiting for a reply. If a reply is received it is presumed that the other node is still available and no problems are present. However if no reply is
received an error in the other node is presumed. This tool was developed for providing highly available clusters. Therefore when we use it with server1 (primary node) and server 2 (secondary node) we can implement high availability i.e. when the primary node fails the signal to the secondary server will not arrive, therefore the secondary server will take it’s place in providing storage to the network.

We will now install and configure these packages on server1 and server2 through a ssh connection from the terminal in oneserver. All features are to be installed on both servers unless specifically stated.

1. Power on server1, server2 and oneserver
2. Open two terminal windows in oneserver and ssh to server1 and server2 (as below). Accept the host and enter the user passwords you set

```
Terminal1 : ssh localadmin@server1
Terminal2 : ssh localadmin@server2
```

a.

Network File Server and replication setup

3. We now have to install ntp and ntpdate in order to syncronise time. This will help in communication and logs

```
Sudo apt-get install ntp ntpdate
```
4. The next step is to install the network file storage server. This will help provide access to storage throughout the network.

   `sudo apt-get install nfs-kernel-server`

5. The network fileserver will be controlled by the heartbeat package therefore we must remove it from the startup. Enter the following command twice on both servers. It should return the values below.

   `sudo update-rc.d -f nfs-kernel-server remove`

6. You must now edit the `/etc/exports` file in both servers to make the folder “/var/lib/one/” available over the network.

   ```
sudo pico /etc/exports

# insert the following line

/var/lib/one/ 192.168.1.0/255.255.255.0(rw,no_root_squash,no_all_squash,sync)
```

7. You must now install and configure `drbd8` as discussed earlier.

   `sudo apt-get install drbd-utils drbdlinks`

8. Now we must change `drbd` configuration (`/etc/drbd.conf`) to suit our installation. 
   e.g. set storage devices, synchronisation rate and error handling.
resource r0 {
    protocol C;
        handlers { pri-on-incon-degr "halt -f"; }
    startup {
        degr-wfc-timeout 120; ## 2 minutes.
    }
    disk {
        on-io-error detach;
    }
    Net {
    }
    syncer {
        rate 10M;
        al-extents 257;
    }
    on server1 {
        device /dev/drbd0;
        disk /dev/sda8;  
            # Data partition on server 1
        address 192.168.1.94:7788;  
            # IP address on server 1
        meta-disk /dev/sda7[0];  
            # Metadata for DRBD on server 1
    }
    on server2 {
        device /dev/drbd0;
        disk /dev/sda8;  
            # Data partition on server 2
        address 192.168.1.97:7788;  
            # IP address on server 2
        meta-disk /dev/sda7[0];  
            # Metadata for DRBD on server 1
    }
}

in Server 1&2 : load the DRBD kernel module

9. When the configuration is set we then must load the drbd kernel

    sudo modprobe drbd
10. Now we must create the meta-disk for drbd. This is basically a partition used to store meta data i.e. data about data – the who, what where and when information about how data is dealt with within drbd. We created the partition of 200MB during the installation.

```
sudo dd if=/dev/zero of=/dev/sda7 bs=1M
sudo count=128
```

11. We can now start drbd. This will start synchronisation between the servers. You can test this by running the second command. The expected output is below.

```
sudo drbdadm up all
sudo cat /proc/drbd
```

a. If that message is shown its good news as it means they are connected, if not check Check your configuration files.

12. The next step is to set your primary and secondary server. The primary server will sync its files to the secondary. To do this run the following command on the primary server ONLY(server1 in this case)

```
sudo drbdadm --overwrite-data-of-peer primary all
sudo drbdadm disconnect r0
sudo drbdadm -- connect all
```

a. Now the primary server will start syncing with the secondry. To test and see its progress run:

```
sudo cat /proc/drbd
```

b. The output shows the synchronisation progress, transfer speed and time remaining.

13. The last steps in configuring the network file storage is making the NFS folder identicle in both servers to ensure both behave identically in case of failure. To do this you must first make the folder /var/lib/one. This is where the Opennebula drivers and datastores will be stored.
a. Then you must mount the shared partition to /var/lib/one and then move the nfs folder to /var/lib/one. You must then link /var/lib/one/nfs with /var/lib/nfs.

i. These commands should only be carried out in server1 only

```
sudo mkdir /var/lib/one

sudo mount -t ext3 /dev/drbd0 /var/lib/one
sudo mv /var/lib/nfs/ /var/lib/one
sudo ln -s /var/lib/one/nfs/ /var/lib/nfs
sudo umount /var/lib/one
```

b. Then **you must make changes** to the NFS in **server2**

```
rm -fr /var/lib/nfs/
ln -s /var/lib/one/nfs/ /var/lib/nfs
```

Heartbeat

1. The first step is to install the heartbeat package

```
sudo apt-get install heartbeat
```

2. When heartbeat is installed we create a configuration file (/etc/heartbeat/ha.cf) specifying different attributes such as hosts and logs

```
sudo pico /etc/heartbeat/ha.cf
#insert into ha.cf

logfacility local0
keealive 2
deadtime 10
bcast eth0
node server1 server2
```

3. The next step is adding high availability resources to the /etc/heartbeat/haresources file. Here we define the shared folder

```
sudo pico /etc/heartbeat/haresources
#add the following line to the file, save and then exit

server1 IPaddr::192.168.1.174/24/eth0 drbdisk::r0 Filesystem::/dev/drbd0::/var/lib/one::ext3 nfs-kernel-server
```
4. The next step is to change permissions on the heartbeat authkeys folder so no users can access it without permission

```
sudo chmod 600 /etc/heartbeat/authkeys
```

5. Now we will start the drbd and heartbeat services

```
sudo service drbd start
sudo service heartbeat start
```

onenode setup

onenode is used with the xen hypervisor to run the virtual machines. We installed the operating system earlier so now we will move on to configuring the operating system and install the Xen hypervisor. But first we must add the network bridge for xen and add the “oneadmin” user to the system for Opennebula.

We configured the network already during setup so now we will configure the “xenbr0” bridge. Xenbr0 is the default xen network bridge. A network bridge allows the vms connect to the network using the hosts IP address. To configure the bridge we use the package “bridge-utils” which is a tool which allows administrators create and manage bridged networks.

1. To install bridge utils we run the command

```
sudo apt-get install bridge-utils
```
2. After the package installs we will edit the network interface (/etc/network/interfaces) as we did with oneserver.

```bash
sudo pico /etc/network/interfaces
```

- We then edit the file so it looks like this

```plaintext
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet manual
auto xenbr0
iface xenbr0 inet static
    address 192.168.1.97
    netmask 255.255.255.0
    network 192.168.1.0
    broadcast 192.168.1.255
    gateway 192.168.1.1
dns-nameservers 192.168.1.1
dns-search xencloud.com
bridge_ports eth0
bridge_fd 9
bridge_hello 2
bridge_maxage 12
bridge_stp off
```

- We must then add our hosts to “/etc/hosts” file. Please make sure the following hosts are added.

```bash
sudo pico /etc/hosts
```

```plaintext
192.168.1.96    oneserver.xencloud.com    oneserver
192.168.1.97    onehost.xencloud.com    onehost
```

i. We should then reboot the machine to make changes or you could run the command

```bash
sudo /etc/init.d/networking restart
```
3. When the system reboots we can add the “oneadmin” user. This is the account used by Opennebula for administration purposes. It is important that you follow these steps carefully as wrong configuration can cause errors later on in the installation.
   
a. Firstly you add the group oneadmin with the id “10000”
   
   ```
   sudo groupadd -g 10000 oneadmin
   ```
   
b. Next you add user “oneadmin” with the id “10000” to the group “oneadmin” with the folder “/var/lib/one/” as its home directory
   
   ```
   sudo useradd -u 10000 -m oneadmin -d /var/lib/one -s /bin/bash -g oneadmin
   ```
   
c. Now we must set a password for oneadmin
   
   ```
   sudo passwd oneadmin
   ```
   
d. We must set ownership of the “/var/lib/one/” folder to user and group oneadmin
   
   ```
   sudo chown -R oneadmin:oneadmin /var/lib/one
   ```
   
e. Now test that your login works and then exit again to the “localadmin” user.
   
   ```
   su -l oneadmin
   exit
   ```

4. The next step is to install, configure and mount the network file storage. To do this we use the package “nfs-common”.
   
a. To install this package
   
   ```
   sudo apt-get install nfs-common
   ```
   
b. Now we must configure the machine to use the NFS from the replication server. To do this we must edit the file “/etc/fstab”.
   
   ```
   sudo pico /etc/fstab
   #insert the following, save and exit
   192.168.1.174:/var/lib/one /var/lib/one nfs rw,vers=3 0 0
   ```
   
c. Now we will create the directory “/var/lib/one/” if it does not exist and mount it to the network file storage. The last command mount should show as `the network storage has been mounted successfully.
   
   ```
   sudo mkdir /var/lib/one/
   sudo mount /var/lib/one
   mount
   ```
Setup the Xen Hypervisor

1. Now that the node has been fully configured we can install the Xen hypervisor. To do this we run the following commands
   a. To install the hypervisor
      
      ```
      sudo apt-get install xen-hypervisor-amd64
      ```
   b. You must then set the grub to boot xen as the default option and run the command update grub in order to save the changes
      
      ```
      sudo sed -i 's/GRUB_DEFAULT=.*/GRUB_DEFAULT="Xen 4.1-amd64"/ /etc/default/grub
      sudo update-grub
      ```
   c. You must then set the default toolstack for xen as “xm” as that is what is used with Opennebula. Also add a limit to how cpu and memory Xen can use by editing the grub. Change appamor=”” to appamor=”0” and add the following line in, instead of grub default GRUB_CMDLINE_XEN="dom0_mem=2G,max:2G dom0_max_vcpus=2"
      
      ```
      sudo sed -i 's/TOOLSTACK=.*/TOOLSTACK="xm"/ /etc/default/xen
      sudo update-grub
      sudo pico /etc/default/grub
      GRUB_CMDLINE_XEN="dom0_mem=2G,max:2G dom0_max_vcpus=2"
      #save and exit
      sudo update-grub
      sudo reboot
      ```
   
   from the frontend. Therefore ruby will need to be installed

      ```
      sudo apt-get install ruby
      ```
   
   i. It is also important to note additional requirements if you are running xen 4.1
   
   1. When running a vm Opennebula points to the wrong folder for keymaps. A fix to this is creating a symlink using the following command
      
      ```
      ln -s /usr/share/qemu-linaro/keymaps /usr/share/qemu/
      ```
2. You will also need to grant the oneadmin user password less sudo commands in order to run scripts from Opennebula. In order to do this you edit the sudoers file as localadmin as follows.

```
sudo visudo
#add the following two lines, save and exit
%xen    ALL=(ALL) NOPASSWD: /usr/sbin/xm *
%xen    ALL=(ALL) NOPASSWD: /usr/sbin/xentop *
```

2. Now that Xen is installed the node is nearly fully configured. We just need to make sure openssh-server is installed to allow Opennebula connect to the node.

```
sudo apt-get install openssh-server
```

Now that the onenode is fully configured we can move back and configure oneserver – the “front-end” and start getting operational!

**oneserver frontend setup**

1. As with the previous node - “onenode” we add the “oneadmin” user. This is the account used by Opennebula for administration purposes. It is important that you follow these steps carefully as wrong configuration can cause errors later on in the installation.

   a. Firstly we create the “/var/lib/” folder if it has not been previously created on this node

```
sudo mkdir -p /var/lib
```

   b. Then you add the group oneadmin with the id “10000”

```
sudo groupadd -g 10000 oneadmin
```

   c. Next you add the user “oneadmin” with the id “10000” to the group “oneadmin” with the folder “/var/lib/one/” as its home directory

```
sudo useradd -u 10000 -m oneadmin -d /var/lib/one -s /bin/bash -g oneadmin
```

d. Now we must set a password for oneadmin

```
sudo passwd oneadmin
```

e. We must set ownership of the “/var/lib/one/” folder to user and group oneadmin
f. Now test that your login works and then exit again to the “localadmin” user.

```
su -l oneadmin
exit
```

2. The next step is to install, configure and mount the network file server. To do this we use the package “nfs-kernel-server”.

a. To install this package

```
sudo apt-get install nfs-kernel-server
```

b. Now we must configure the machine to use the NFS from the replication server so that all files created reside on the network file storage and not on local storage. To do this we must edit the file “/etc/fstab”.

```
sudo pico /etc/fstab

#insert the following, save and exit

192.168.1.174:/var/lib/one /var/lib/one nfs rw,vers=3 0 0
```

c. Now we must edit the “/etc/exports” folder to make “/var/lib/one” accessible to other nodes (where “192.168.1.97” is IP of onenode.)

```
sudo pico /etc/exports

#add the following line, save and exit

/var/lib/one 192.168.1.97(rw, sync, no_subtree_check, no_root_squash, anonuid=10000, anongid=10000)
```

d. For the last step we must start the Network File server. It should return the status from the picture below. We can also list the storage devices using “df –h” to confirm the directory is mounted.

```
sudo /etc/init.d/nfs-kernel-server start

df -h
```
3. The next step is to set up password-less secure shell access to the onenode. To do this we must log in as oneadmin and generate our public and private keys. When prompted to enter a passphrase and directory for the ssh keys just press enter i.e. choose the defaults.

   Note: You must be logged in as oneadmin

   su -l oneadmin
   ssh-keygen

   a. When the keys has been successfully generated we need to copy the oneserver public key into the “authorized_keys” file in order to allow passwordless access. To do this use the following command.

   cat ~/.ssh/id_rsa.pub > ~/.ssh/authorized_keys

   b. The next step is to create a configuration file (/var/lib/one/.ssh/config) for ssh. Here we tell openssh to accept requests from all hosts without strict host key checking

   Pico ~/.ssh/config
   #add the following lines to the file, save and exit
   Host *
   StrictHostKeyChecking no

   Now that all nodes are configured correctly it is time to install Opennebula on oneserver

Opennebula Installation

There are two ways you can install Opennebula

- Through the Ubuntu package repository
- Through Source

We have chosen to install from source as it is the latest version and therefore better support is available and all known bugs have been patched. The Ubuntu repository takes a while to be updated. Opennebula version 4 is in the repository while the latest stable release available from source is Opennebula 3.8.

We will retrieve the source code from the official Opennebula github repository, compile it and then install it.

1. To retrieve the source from github we will need two packages – “git” and “git-core”.

   sudo apt-get install git git-core
2. When these packages are installed we can now clone the github repository to our “oneserver” node. I recommend you create a dedicated folder for the Opennebula source in case you need to roll back to an earlier version and re-compile/re-install. Therefore we will create a dedicated directory.

```
sudo mkdir /var/lib/opennebula_source/
sudo chown -R oneadmin:oneadmin /var/lib/opennebula_source/
sudo mkdir /var/lib/opennebula_source/opennebula_3.8
cd /var/lib/opennebula_source/opennebula_3.8
```

3. Now that we have created the folder structure and change location to the specified folder we can login as oneadmin and clone the source code from Github.

```
Su –l oneadmin
cd /var/lib/opennebula_source/opennebula_3.8
git clone https://github.com/OpenNebula/one
exit
```

a. Before we compile and install the source there are a number of packages we need to install in order to carry out this task and run Opennebula

```
cd /var/lib/opennebula_source/opennebula_3.8
sudo apt-get install gitlibcurl3 libmysqlclient18 libruby1.8 libsqlite3-ruby libsqlite3-ruby1.8 libxmlrpc-c3-dev libxmlrpc-core-c3 mysql-common ruby ruby1.8
sudo apt-get install libxml2-dev libmysqlclient-dev libmysql++-dev libsqlite3-ruby libexpat1-dev
sudo apt-get install libc6 libgcc1 libmysqlclient18 libpassword-ruby libsequel-ruby libsqlite3-0 libssl0.9.8 libstdc++6
libxml2
sudo apt-get install ruby rubygems libmysql-ruby libsqlite3-ruby libamazonec2-ruby
sudo apt-get install libsqlite3-dev libxmlrpc-c3-dev g++ ruby libopenssl-ruby libssl-dev ruby-dev
sudo apt-get install libxml2-dev libmysqlclient-dev libmysql++-dev libsqlite3-ruby libexpat1-dev
sudo apt-get install rake rubygems libxml-parser-ruby1.8 libxslt1-dev genisoimage scons
sudo gem install nokogiri rake xmlparser
sudo apt-get install mysql-server
```
b. One the packages are installed we have to configure the mysql server. We must add the oneadmin user and grant all privileges to it by entering the mysql shell and executing the following queries

```
mysql -uroot -pgalway123
CREATE USER 'oneadmin'@'localhost'
IDENTIFIED BY 'oneadmin';

CREATE DATABASE opennebula;

GRANT ALL PRIVILEGES ON opennebula.* TO 'oneadmin' IDENTIFIED BY 'oneadmin';
quith;
```

c. Now we can compile and install Opennebula using some of the packages just installed. We must make sure we execute the commands as root from the source codes folder.

```
 cd /var/lib/Opennebula_source/Opennebula_3.8
sudo scons sqlite=no mysql=yes
sudo ./install.sh -u oneadmin -g oneadmin -d /var/lib/one
```

Now Opennebula is installed! But we have to do a couple of configuration changes before we can become operational

**Configuring Opennebula**

Before starting Opennebula we must

1. create a profile file defining environmental variables for Opennebula’s drivers. These variables are used to point towards certain Opennebula folders.

```
su –l oneadmin
pico ~/.bash_profile
#add the following to the file, save and exit
export ONE_LOCATION=/var/lib/one
export ONE_AUTH=$ONE_LOCATION/.one/one_auth
export ONE_XMLRPC=http://localhost:2633/RPC2
exportPATH=$ONE_LOCATION/bin:/usr/local/bin:/var/lib/gems/1.8/bin:/var/lib/gems/1.8/:$PATH
```
2. Once you have created the file and saved it you should execute the file and test it using echo (it should return /var/lib/one) to see if its successful.

```bash
source ~/.bash_profile
echo $ONE_LOCATION
```

3. Now you must store the Opennebula username and password in the .one auth file. This will be your Opennebula username and password and is different to your Ubuntu Opennebula password. You must also set ownership to oneadmin so unauthorised access and editing cannot take place.

```bash
mkdir ~/.one
echo "oneadmin:<TYPE THE PASSWORD HERE>" > ~/.one/one_auth
sudo chown oneadmin:oneadmin /var/lib/one/.one/one_auth
```

4. We must now edit the oned configuration file to make Opennebula suit our installation. By default opennebula activates KVM drivers in this file. We must comment these out and uncomment the Xen drivers i.e. remove “#” on xen drivers while inserting “#” before each KVM driver. We also need to change the database driver from sqlite to mysql (below).

```bash
su -l oneadmin
pico ~/etc/oned.conf
#DB = [ backend = "sqlite" ]
b. Set SQL as MYSQL-uncomment
#lines 61 through 66 or near by
DB = [ backend = "mysql",
      server = "localhost",
      port = 0,
      user = "oneadmin",
      passwd = "oneadmin",
      db_name = "opennebula" ]
```

5. Now that one is configured, log in as oneadmin and you can start opennebula using the command

```bash
su -l oneadmin
one start
```
**Using Opennebula**

Now that we have configured Opennebula we can add, delete and monitor hosts. We will start by adding the host “onenode”. To do this we make sure Opennebula is running and if it is we can add onenode to the cluster

```
    su -l oneadmin
    onehost create onenode --im xen --vm vmm_xen --net dummy
```

If the host was added correctly you will be given an id.

```
    oneadmin@oneserver:~$ onehost create onenode --im xen --vm vmm_xen --net dummy
    ID: 5
```

With this id you can monitor the host. You can also monitor the host using the “onehost list” or “onehost top” command.

```
    onehost top
    onehost show 5
```

To demonstrate creating a virtual network, adding an image to the repository, adding a template and deploying a virtual machine we will use the ttyl linux operating system. Ttylinux is a small linux distribution ideal for our environment due to lack of resources.
**Launching a virtual machine**

The first step is retrieving the required image and storing it in oneserver.

1. To do this we create a folder for storing it and then download it to that folder

   ```
   mkdir /var/lib/image_templates/ttylinux
   cd /var/lib/image_templates/ttylinux
   wget http://dev.opennebula.org/attachments/download/355/ttylinux.tar.gz
   ```

2. Now we must untar the image using the following command

   ```
   tar xvf ttylinux.tar.gz
   ```

3. After the file unpacks we must create a network, image and deployment template

Here are the examples I used

   a. **network.net**

   ```
   pico network.onenet
   NAME = "my first network"
   TYPE = FIXED
   BRIDGE = xenbr0
   LEASES = [ IP="192.168.66.5"]
   ```

   ```
   onevnet create network.net
   ```

   b. **image.one**

   ```
   pico image.one
   NAME = ttylinux
   PATH = "/var/lib/image_templates/ttylinux/ttylinux.img"
   TYPE = OS
   PUBLIC = YES
   DESCRIPTION = "ttyl image"
   ```

   i. In order to create the image we use the “oneimage” command and it creates the image in the Opennebula datastore. You can select which datastore based on the “–d” tag. When the image creates it will give you an id

   ```
   oneimage create image.one –d default
   ```
c. myfirsttemplate.one

```ruby
CPU="1"
DISK=[ BUS="ide", IMAGE="ttylinux ", IMAGE_UNAME="oneadmin", READONLY="no", TARGET="xvda" ]
GRAPHICS=[ LISTEN="0.0.0.0", TYPE="vnc" ]
MEMORY="512"
NAME="my first vm"
NIC=[ NETWORK="my first network ", NETWORK_UNAME="oneadmin" ]
OS=[
  KERNEL="/usr/lib/xen-default/boot/hvmloader" ]
RAW=[
  DATA="builder = 'hvm'
  shadow_memory = 8
device_model = '/usr/lib/xen-default/bin/qemu-dm'
  boot = "c",
  TYPE="xen" ]
REQUIREMENTS="HOSTNAME = \"onehost\"
TEMPLATE_ID="1"
```

i. In order to create the template we use the “onetemplate” command and it creates the template in Opennebula ready to deploy the vm. We then use the command “onetemplate instantiate <template id>” to instantiate the vm.

```
onetemplate create myfirsttemplate.one
onetemplate instantiate 1
```

The vm should now be running. You can check its status using the command

```
onetvm top
```

You now have a working private cloud. However to make life easier for administrators and users Opennebula also has a self-service web portal used for managing and monitoring infrastructure and it is called the sunstone server.

**Installing Sunstone self service portal**

To install sunstone you do the following

1. Firstly, to run sunstone you need to install the gems nessesary to run the server.

```
sudo "~/share/install_gems sunstone
```
2. When the installation finishes you can then install novnc using the following command

```
    cd ~/share
    sudo ./install_novnc.sh
```

3. Sunstone is now ready to run! Simply enter the following command and then open firefox/chromium and navigate to http://127.0.0.1:9869

```
    Sunstone-server start
```

When logged in to sunstone the user can use all of opennebula’s features though the very attractive gui provided. It also provides a very valuable monitoring tool.

In this example of monitoring, this is a great tool as it shows a lot. It shows the amount of memory which has been allocated, the cpu usage, if there is any additional memory for your VM’s. we are shown how many users there are per group and also how many groups there are.
We can also see in an easier way, the different VM’s available to us including if they are running properly and who has rights to them. The ip’s are shown and also the log associated with that VM which will show us step-by-step what is wrong with the VM if there is an error. There is a chart showing how much memory is being used or if it is being overloaded or if the network is being overused.

This is an example of monitoring in Xen, here we have sufficient information about our VM such as the name, size, state and owner of each VM which is created. We can find out when they have been created, along with that we can show information as regards the network. There we can see where it belongs too and the type of network that it is. We can also see the templates which we created. To see the images and templates we created, we used the command ‘oneimage host’ to see the images and ‘onetemplate list’ to see the templates.
Provisioning of the public cloud

This section details the steps taken to launch an instance on amazon web services through our local Opennebula infrastructure. AWS drivers are built into Opennebula therefore making implementation a lot easier for users and provide infinite scalability. These factors make hybrid cloud computing with Opennebula an attractive proposition.

We have not implemented a public cloud integration yet but have research the topic and we have found that the following is the procedure to follow in configuring Opennebula with AWS.

1. Register an account with amazon web services
To register with amazon web services you simply go the http://aws.amazon.com, click register and enter your details including credit card information for payment and identification reasons. When you are fully registered log in and navigate to the aws console.

2. Set up ssh keys on public cloud
Enter the key section in the aws management console select import keypair. This will open a prompt allowing you to upload the public key of your front-end. This will allow you to access future instances remotely.

3. Generate x.509 certificate to authenticate openenbula with AWS
In order to use the remote amazon api’s we need to generate a X.509 certificate to authenticate with AWS. To do this we need to access the security credentials section of your AWS account (https://aws-portal.amazon.com/gp/aws/securityCredentials). When at this page select “access credentials and select the X.509 certificate tab. Generate a new certificate and download it to your “/var/lib/one/etc” and change permissions on the file so only “oneadmin” can access it.

4. Download the aws api
Before starting download the Java JRE package on the frontend as it is required by AWS api tools.

   sudo apt-get install openjdk-6-jre-headless

Edit your profile file to include the following line “export JAVA_HOME=/usr/lib/jvm/java-6-openjdk” and then load the source.

Now download the amazon api tools from http://aws.amazon.com/developertools/351 and save them in a folder on the front-end e.g /var/lib/one/ec2-api-tools.

5. Then we must configure Opennebula drivers and set the EC2 information manager configuration
The next step is to uncomment the AWS drivers in the openenbula configuration file and reboot the frontend. (oned.conf)

Also in the /var/lib/one/etc/im_ec2/im_ec2.conf file we can set the instance types and numbers permitted. This is a very important feature as you do not want to over provision resources as it can become very costly.

6. EC2 API Tools configuration
Now we must tell opennebula where the aws api tools, certificate and keys are stored. To do this we edit the “/var/lib/one/etc/vmm_ec2/vmm_ec2rc” file and change the relevant paths to where we
have stored the keys, api and cert. It is also good practice to assign environment variables to your credentials and the api to give ease of access. To do this we edit the profile file and insert the following.

```plaintext
export EC2_HOME=$ONE_LOCATION/share/ec2-api-tools
export EC2_PRIVATE_KEY=$ONE_LOCATION/etc/pk.pem
export EC2_CERT=$ONE_LOCATION/etc/cert.pem
PATH=$EC2_HOME/bin:$PATH
```

7. Add AWS host to Opennebula
One all those settings are modified we should be able to add a AWS host the same way as we would a normal local host except using the AWS drivers.

```plaintext
onehost create ec2 im_ec2 vmm_ec2 tm_dummy dummy
```

We have also research monitoring tools for the public cloud and have found a feature called cloud watch which is a service provided by amazon cloud watch which when configured right can return stats on availability and status. But we cannot directly monitor the vms on AWS as there is no direct access to the hosts on AWS for security reasons.

8. Launch instance
Firstly we need to define a vm template like before. Which will contain only the the instance type and the uniquie AWS image identifier (AMI). Every image on aws is assigned a AMI and it is a very good way of identifying different images. The following is an example image template named myfirstAWS.one

```plaintext
EC2 = [
    AMI= "ami-4a0df923",
    INSTANCETYPE=t1.micro
]
```

Then we must instantiate the image using the usual command of onevm. To check if the deployment was successful run the onevm show command as below, with the image id. It should return the information on the vm including the IP address. If it does the vm has been successfully created.

```plaintext
onevm create myfirstAWS.one
onevm show <imageid>
```
Conclusion
In conclusion we have set up a private cloud infrastructure which has the capability to burst onto AWS.