Open Nebula 3.4.1

Setup a HIGH-AVAILABILITY environment using Ubuntu 12.04, MySQL, DRBD and Heartbeat

OpenNebula VM : Windows XP

INDEX

By

ANIL KUMAR

cloud.b.lab@zoho.com

www.cloud-b-lab.com

www.cloud-b-lab.co.in

Chennai and Trivandrum , India

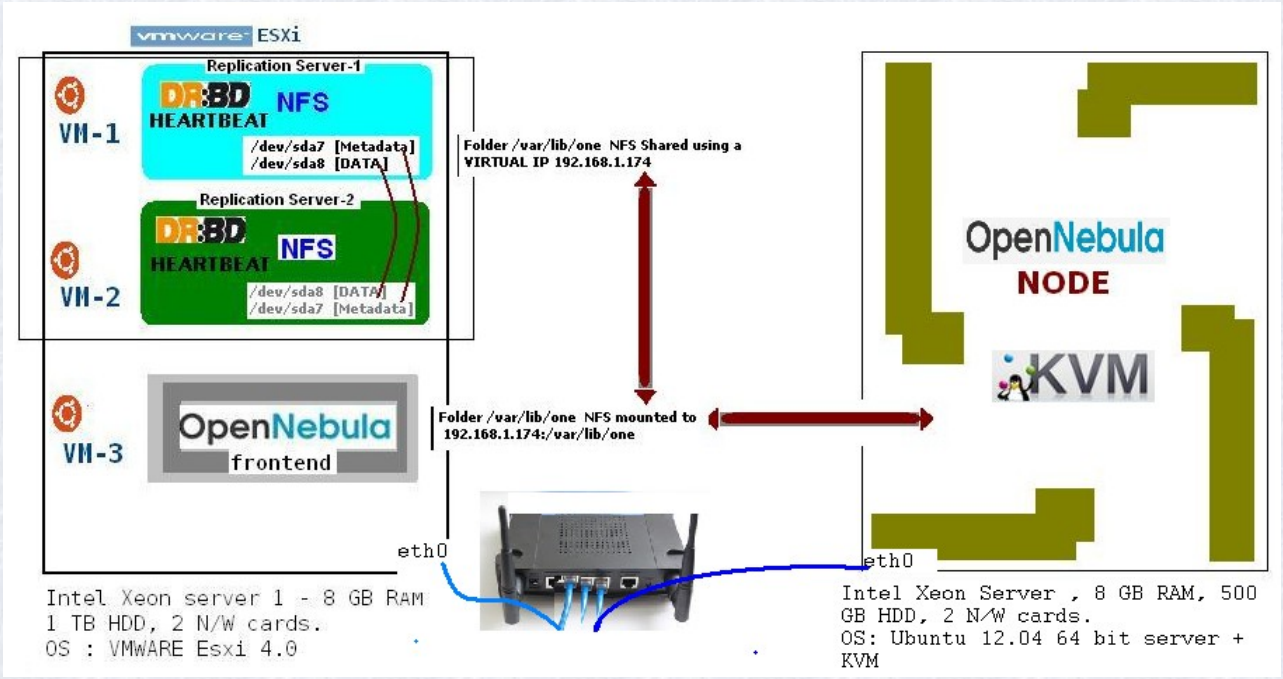
Friends, this tutorial teaches you to setup a OpenNebula 3.4.1 HIGH-AVAILABILITY environment but with minimum hardware resources and some easy steps.

"High availability" refers to the practice of keeping online resources available through node failure or system maintenance.

I am using just 2 VT enabled Intel Xeon Servers and a normal Intel dual core Windows XP laptop with 2GB Ram for having remote sessions and VSphere client.

MySQL,DRBD and Heartbeat packages already available with Ubuntu 12.04 will be used, so that installation and setup will be a bit easy.

Now, let us have a view at how a completed setup would look like:



As a case:OpenNebula server folder "*/var/lib/one*" is made highly available.

That means folder ***/var/lib/one*** will be made available between replication server 1 and server2. Same folder will be exposed out using s NFS share so that OpenNebula front end and OpenNebula Node together can share it.

Requirements

1. VMWARE Server : A VT Enabled server machine
2. OpenNebula Node : A VT Enabled machine
3. A Windows/Linux Laptop/Desktop with VSphere Client , putty etc

Preparing Replication Servers:

1. Install VMWARE ESX/ESXi
2. Create 3 VMS and install Ubuntu 12.04 64 bit server in all 3 VMs
3. 2 Ubuntu VMs will be used to setup our replication environment.
4. 1 Ubuntu VM will be prepared as OpenNebula Front end.
5. During Ubuntu installation, once you set the host name and select eth0 as Primary n/w, while DHCP process runs, press cancel button, which will enable you to enter IP address ,netmask, gateway manually.
6. All Vms to have at least 1GB RAM
7. **VM-1**

Hostname	**Server1**	
IP eth0	**192.168.1.94**	
Gateway	**192.168.1.1**	
Domain/DNS	**Example.com / 192.168.1.1**	
HDD Partitioning		
Partition-1	**100 MB - ext3 - Boot - /dev/sda1**	
Partition-2	**5 GB - etx3 – Root - /dev/sda5**	
Partition-3	**1 GB – Swap - /dev/sda6**	
Partition-4	**200 MB - ext3 - NO MOUNT -/dev/sda7**	**For DRDB Metadata**
Partition-5	**60 GB - etx3 - NO MOUNT -/dev/sda8**	**For DRDB DATA**
Username	**localadmin**	
Additional pkg	**Openssh server**	

- **VM-2**

Hostname	**Server2**	
IP eth0	**192.168.1.97**	
Gateway	**192.168.1.1**	
Domain/DNS	**Example.com / 192.168.1.1**	
HDD Partitioning		
Partition-1	**100 MB - ext3 - Boot - /dev/sda1**	
Partition-2	**5 GB - etx3 – Root - /dev/sda5**	
Partition-3	**1 GB – Swap - /dev/sda6**	
Partition-4	**200 MB - ext3 - NO MOUNT -/dev/sda7**	**For DRDB Metadata**
Partition-5	**60 GB - etx3 - NO MOUNT -/dev/sda8**	**For DRDB DATA**
Username	**localadmin**	
Additional pkg	**Openssh server**	

- **VM-3**

Hostname	**OneServer**	
IP eth0	**192.168.1.200**	

Gateway	**192.168.1.1**	
Domain/DNS	**Example.com / 192.168.1.1**	
HDD Partitioning		
Partition-1	**Single Partition wth whatever size [say 60 GB]**	
Username	**localadmin**	
Additional pkg	**Openssh server**	

8. Preparing **Server1 and Server2 [hereafter specified as Server 1&2] for** data replication

a) Open two putty/SSH terminal windows connecting to Server1 and Server2

b) In **Server 1&2** – install ntp , ntpdate for time synchronization

```
apt-get install ntp ntpdate
```

c) In **Server 1$2** – install NFS server

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

d) We should remove the system bootup links for NFS because NFS will be started and controlled by *heartbeat* automatically. Execute below instructions in **Server 1&2**

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

e) Using NFS share, export the directory */var/lib/one. Edit /etc/exports and add below line in* ***Server 1&2****, save and exit*

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

Install DRBD:

As per http://www.drbd.org/: DRBD® refers to block devices designed as a building block to form high availability (HA) clusters. This is done by mirroring a whole block device via an assigned network. DRBD can be understood as network based raid-1.

f) *in* ***Server 1&2*** *: Install DRBD*

```
apt-get install drbd8-utils  drbdlinks
```

g) *in* ***Server 1&2*** *: edit* ***/etc/drbd.conf*** *and save below configuration info.*

```
# You can find an example in  /usr/share/doc/drbd.../drbd.conf.example
#include "drbd.d/global_common.conf";
#include "drbd.d/*.res";
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 2
  }
}
```

h) *in* ***Server 1&2 :*** *load the DRBD kernel module*

```
modprobe drbd
```

i) *in* ***Server 1&2 : create a drbd – meta disk. Before that we need to zero size the /dev/sd7 partition in both servers***

```
dd if=/dev/zero of=/dev/sda7 bs=1M count=128
drbdadm create-md r0
drbdadm up all
cat /proc/drbd
```

Cat /prod/drbd will show something like below in both servers.

0: cs:Connected st:Secondary/Secondary ld:Inconsistent
ns:0 nr:0 dw:0 dr:0 al:0 bm:1548 lo:0 pe:0 ua:0 ap:0

j) ***In Server 1 alone : Make server 1 Primary***

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

k) The drbdadm -connect all will start the synchronization of /dev/sd8 between server 1 and sever 2. It will take some time (may be in hours) beased on the size of data partition.

l) *You can monitor the Progress using* ***cat /proc/drbd*** *command.*

Server1:

0: cs:SyncSource st:Primary/Secondary Ds:UpToDate/Inconsistent
ns:13441632 nr:0 dw:0 dr:13467108 al:0 bm:2369 lo:0 pe:23 ua:226 ap:0
[==========>.....] sync'ed: 53.1% (11606/24733)M
finish: 1:14:16 speed: 2,644 (2,204) K/sec

Server2:

0: cs:Connected st:Secondary/Primary Ds:*UpToDate Inconsistent*
ns:37139 nr:0 dw:0 dr:49035 al:0 bm:6 lo:0 pe:0 ua:0 ap:0
[==========>.....] …….

Once finished syncing:

Server 1 : *0: cs:Connected ro:Primary/Secondary ds:UpToDate/UpToDate C r-----*
ns:116 nr:0 dw:116 dr:741 al:5 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f oos:0

Server 2: *0: cs:Connected ro:Secondary/Primary ds:UpToDate/UpToDate C r-----*
ns:0 nr:116 dw:116 dr:0 al:0 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f oos:0

Additional NFS Configuration

m) *in **Server 1&2 :** Make /var/lib/nfs folder information same in both the servers, so that if sever1 goes down server2 will behave identical to server1*

```
mkdir /var/lib/one
```

n) **In Server 1 alone :**

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

o) **In Server 2 alone :**

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

Install And Configure heartbeat:

Heartbeat is very flexible and powerful. A Heartbeat setup can have basic active/passive clusters with two members, where the active server is providing the services and the passive server is waiting to take over if necessary.

p) *Heartbeat will be installed in **Server1&2. It** monitors the other server constantly. For example, if server1 goes down, heartbeat on server2 detects this and makes server2 take over. heartbeat also starts and stops the NFS server on both server1 and server2. It also provides NFS as a virtual service via the IP address 192.168.1.174 so that the openNebula server and node(s) see only one NFS server.*

In Server1&2

```
apt-get install heartbeat
```

q) ***Create configuration files for HeartBeat → In Server 1&2***

1: etc/heartbeat/ha.cf

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

2: /etc/heartbeat/authkeys {change <REPLACE_WITH_A_PASSWORD> with a simple passsword}

```
auth 3
3 md5    <REPLACE_WITH_A_PASSWORD>
```

3: /etc/heartbeat/haresources: Below line remains same in both Server1 and Server2

```
server1  IPaddr::192.168.1.174/24/eth0 drbddisk::r0 Filesystem::/dev/drbd0::/var/lib/one::ext3 nfs-kernel-server
```

r) ***In Server 1&2 :*** *Make /etc/heartbeat/authkeys* should be readable by root

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

s) ***In Server 1&2 :*** *start DRBD and heartbeat on server1 and server2:*

```
sudo service drbd start
```

```
sudo service heartbeat start
```

t) Test it with ifconfig: You will see a new eth0:0 in Server 1 with IP as 192.168.1.174

Server1 :

```
root@server1:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:c0:69:b7
          inet addr:192.168.1.94  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fec0:69b7/64 Scope:Link.............

eth0:0    Link encap:Ethernet  HWaddr 00:0c:29:c0:69:b7
          inet addr:192.168.1.174  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
   …....
```

Server2 :

```
root@server2:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:f9:65:b0
          inet addr:192.168.1.93  Bcast:192.168.1.255  Mask:255.255.255.0
         …..............

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
        …..............
```

u) Test it with df -h : You will see `/dev/drbd0  64G  8.2G  52G  14% /var/lib/one` , only in Server 1

Server1 :

```
root@server1:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda5       4.6G  1.2G  3.3G  27% /
udev            238M  4.0K  238M   1% /dev
tmpfs            99M  252K   99M   1% /run
none            5.0M  4.0K  5.0M   1% /run/lock
none            246M     0  246M   0% /run/shm
/dev/sda1        92M   30M   57M  35% /boot
/dev/drbd0       64G  8.2G   52G  14% /var/lib/one
```

Server2 :

```
root@server2:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda5       4.6G  1.1G  3.3G  25% /
udev            238M  4.0K  238M   1% /dev
tmpfs            99M  252K   99M   1% /run
none            5.0M  4.0K  5.0M   1% /run/lock
none            246M     0  246M   0% /run/shm
/dev/sda1        92M   30M   57M  35% /boot
```

v) Good, Heartbeat setup is done. We will continue further with Heartbeat , once we have OpenNebula front end and Node ready.

Prepare OpenNebula Frontend and Node

1. Setup a bridge in OneServer: Install bridge-utils using below command

```
sudo apt-get install bridge-utils
```

2. Edit "/etc/network/interfaces" file to add a "bridge". Replace the contents as given in **Table VM-T2 and restart networking**

Sample network setup for VMHost

Table VM-T2:

```
# The loopback network interface
auto lo
iface lo inet loopback


# The primary network interface
auto eth0
iface eth0 inet manual
auto br0
iface br0 inet static
    address 192.168.1.200
      netmask 255.255.255.0
      network 192.168.1.0
      broadcast 192.168.1.255
      gateway 192.168.1.1
      # dns-* options are implemented by the resolvconf package, if installed
     dns-search example.com
  bridge_ports eth0
  bridge_fd 9
  bridge_hello 2
  bridge_maxage 12
  bridge_stp off
```

```
sudo /etc/init.d/networking restart
```

```
/etc/resolv.conf

nameserver 192.168.1.1
search westel.com
```

CONFIGURING OneServer

Note: [either work directly on the server console or connect to OneHost using SSH if you have a third machine with putty or any other SSH client]

1. Create a folder "/var/lib"[if doesn't exist] and create a group named "oneadmin"

```
sudo mkdir -p /var/lib/
sudo groupadd -g 10000 oneadmin
```

2. Create a user "oneadmin" , add user to group "oneadmin" and have /var/lib/one as home folder.

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

3. Setup password for "oneadmin" and make oneadmin owner of "/var/lib/one"

```
sudo passwd oneadmin
sudo chown -R oneadmin:oneadmin /var/lib/one
```

4. Test by logging as user "oneadmin" and exit

```
su -l oneadmin
```

```
exit
```

5. Install Network file Server [NFS]

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

6. Edit /etc/fstab and mount the folder /var/lib/one with 192.168.1.174:/var/lib/one.

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

7. Now /var/lib/one of oneserver will be mapped to 192.168.1.174:/var/lib/one. That means what ever you create within /var/lib/one resides inside the replication server and not locally.

8. Edit /etc/exports and add the following line to make folder /var/lib/one/ [192.168.1.95 is of OpenNebula node]

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

9. . Restart NFS server

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

10. create a SSH key for oneadmin and disable host key checking else make all hostkeys known on the OpenNebula node.

```
su -l oneadmin
```

```
ssh-keygen
{Note - all defaults, and no  passphrase.}
```

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

```
nano ~/.ssh/config
[add below two lines to SSH config file]
Host *
StrictHostKeyChecking no
```

```
exit
```

11. Exit from editor and try ssh OneVMHost, it should connect with no password

INSTALL and CONFIGURE OpenNebula in OneServer.

1. Login to OneServer and download OpenNebula Release 3.4

```
su -l oneadmin
Download Latest release of OpenNebula [opennebula-3.4.1.tar] from  http://downloads.opennebula.org/
```

2. Un-tar the build

```
tar xzf opennebula-3.4.1.tar.gz
cd opennebula-3.4.1/
```

3. Before installing OpenNebula, install all pre-requisite packages

```
sudo apt-get install libcurl3 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   [ set the password when asked. I normally give "mygreatsecret" as the pwd]
```

configure MYSql: <refer below screen shot in case of any doubt)

```
mysql -uroot -pmygreatsecret
CREATE USER 'oneadmin'@'localhost' IDENTIFIED BY 'oneadmin';
CREATE DATABASE opennebula;
GRANT ALL PRIVILEGES ON opennebula.* TO 'oneadmin' IDENTIFIED BY 'oneadmin';
quit;
```

```
oneadmin@OneHost:~/opennebula-3.1.90$ mysql -uroot -pmygreatsecret
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 39
Server version: 5.1.58-1ubuntu1 (Ubuntu)

Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL v2 license

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

mysql> CREATE USER 'oneadmin'@'localhost' IDENTIFIED BY 'oneadmin';
Query OK, 0 rows affected (0.00 sec)

mysql> CREATE DATABASE opennebula;
Query OK, 1 row affected (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON opennebula.* TO 'oneadmin' IDENTIFIED BY 'oneadmin';
Query OK, 0 rows affected (0.00 sec)

mysql> quit;
Bye
oneadmin@OneHost:~/opennebula-3.1.90$
```

[you may drag corners to expand]

4. Before installing OpenNebula, configure mysql support.

```
cd  ~/opennebula-3.1.4 [change your folder to opennebula source]
scons sqlite=no mysql=yes
```

5. Install openebula in /var/lib/one accessible by group oneadmin and as user "oneadmin"

```
./install.sh -u oneadmin -g oneadmin -d /var/lib/one
```

6. Create a profile file[~/.bash_profile] to set ENVIRONMENT VARIABLES required to start and use services rendered by "one"

```
nano ~/.bash_profile
export ONE_LOCATION=/var/lib/one
export ONE_AUTH=$ONE_LOCATION/.one/one_auth
export ONE_XMLRPC=http://localhost:2633/RPC2
export PATH=$ONE_LOCATION/bin:/usr/local/bin:/var/lib/gems/1.8/bin/:/var/lib/gems/1.8/:$PATH
```

7. execute the profile file and set the environment variables

```
source ~/.bash_profile
```

[Note: Anytime you open a new SSH window for OneHost, change user to "oneadmin" and source ~/.bash_profile before issuing any "one" command]

8. Create and store OpenNebula user and password in a file. Substitute <TYPE THE PASSWORD HERE> with value

```
mkdir ~/.one
echo "oneadmin:<TYPE THE PASSWORD HERE>" > ~/.one/one_auth
```

9. Make required changes in OpenNebula configuration file ~/etc/oned.conf

```
nano ~/etc/oned.conf

a. comment following line # Line 58 or near by [c hange if your password  for oneadmin is some different]
#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" ]
```

10. Start Nebula

```
one start                { Note: it should start with no error messages}
```

11. Now You can test OpenNebula services

```
onevm list   - this command should execute with no errors. (The list will be empty for now)
```

```
oneadmin@onevmhost:~/ttylinux$ onevm list
   ID USER     GROUP    NAME        STAT CPU     MEM       HOSTNAME       TIME
```

```
 Now login/move  back / to Terminal windows of Server 1 [192.168.1.94]
check the contents of folder /var/lib/one
You will see , all change made above stored in /var/lib/one.
```

Prepare OpenNebula Node [OneVMHost]

Install Ubuntu Server 12.04 64 bit software in VMHost with following parameters.

Ubuntu 12.04 installation steps are same as OneServer, except the values as given in **Table VM-T1**

Do not set the IP address during installation. Let have DHCP. Even if you set it does not matter. We can use/change it post installation.

Table VM-T1:

Partition	You need at least one dedicated partition [e.g ID: 83 System: linux]with 100+GB of free space. Better go for automated partitioning. In case of specific partition choices, go for manual one.
Hostname	OneVMHost

Private N/W- bridge	**Just setup eth0 with DHCP during install**. Post installation add a bridge **br0** as given below [Table: **Sample network setup in oneVMHost1**]
IP[eth0]	DHCP
Username	localadmin [*or have your chosen one*]
Additional software selection	OpenSSH server alone
POST INSTALLATION N/W SETUP:- BRIDGE	
IP[**br0**]	192.168.1.98 [*or a different one as per your setup*]
Netmask	255.255.255.0
Gateway	192.168.1.1
Domain	westell.com

Once installation is over, login to oneVMHost using server console/SSH. You should be able to ping Onevmhost and connect to internet.

Install bridge-utils using below command

```
sudo apt-get install bridge-utils
```

Edit "/etc/network/interfaces" file to add a "bridge".

Sample network setup for oneVMHost **1**

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet manual
auto br0
iface br0 inet static

        address 192.168.1.95
        netmask 255.255.255.0
        network 192.168.1.0
        broadcast 192.168.1.255
        gateway 192.168.1.1
        # dns-* options are implemented by the resolvconf package, if installed
        dns-nameservers 192.168.1.1
        dns-search westell.com
 bridge_ports eth0
   bridge_fd 9
   bridge_hello 2
   bridge_maxage 12
   bridge_stp off
```

```
sudo /etc/init.d/networking restart
```

Add onevmhost,oneserver to the hosts file [/etc/hosts] of oneVMHost

```
192.168.1.200    oneserver.westell.com  oneserver
192.168.1.95   onevmhost.westell.com onevmhost
```

Add onevmhost to the hosts file [/etc/hosts] of oneserver

```
192.168.1.200   oneserver.westell.com  oneserver
192.168.1.95   onevmhost.westell.com onevmhost
```

CONFIGURING oneVMHost

- Before configuring oneVMHost1 check if you are able to ping OneVMHost and the gateway
- Now try to ping 192.168.1.1 , 192.168.1.97 from oneVMHost1
- Also check the internet connectivity in VMHost by pinging *www.ubuntu.com*

```
Install "NFS common" to enable access to the folder "/var/lib/one" of
OneVMHost. [ if run as "oneadmin" . First add oneadmin to SUDOERS file]
```

```
sudo apt-get update
sudo apt-get install nfs-common
```

Edit /etc/fstab and add an NFS entry for /var/lib/one. You will be using the NFS share from replication Server

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

create folder structure /var/lib/one in VMHost and mount it as per "fstab" entry

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

Create user "oneadmin" , groud oneadmin as you did in "oneHost"

```
sudo groupadd -g 10000 oneadmin
sudo useradd -u 10000 -g oneadmin -m oneadmin -s /bin/bash
sudo usermod -d /var/lib/oneoneadmin
sudo usermod -a -G oneadmin,root oneadmin
sudo passwd oneadmin
sudo chown oneadmin:oneadmin /var/lib/one
sudo mount /var/lib/one
mount
Note: You should see the below line
192.168.1.174:/var/lib/one on /var/lib/one type nfs (rw,vers=4,addr=192.168.1.174)
```

Install KVM hypervisor [it will take around 2 minutes or less based on your internet speed]

```
Sudo apt-get install qemu-kvm libvirt-bin ubuntu-vm-builder bridge-utils ruby
```

Libvirt needs to be configured to enable users of group "oneadmin" to manage the Vms and to allow VNC connections. Edit "/etc/libvirt/libvirtd.conf" and make the following two changes

```
unix_sock_group = "oneadmin"
(Search for string "unix_sock", if commented, uncomment this line and change the existing value to "oneadmin").
```

Edit /etc/libvirt/qemu.conf and uncomment vnc_listen line and restart libvirt

```
vnc_listen = "0.0.0.0"
```

```
sudo service libvirt-bin restart
```

Configure libvirt to allow access from the members of group "oneadmin

```
sudo chown :oneadmin /var/run/libvirt/libvirt-sock
```

That's it. We have configured a host machine. Now lets add this host [onevmhost] to OpenNebula Front end.

Check if password less ssh is possible to onevmhost. You should be able to connect to onevmhost with no password , before adding it to OpenNebula front-end as a new host.

ADMINSTRING OpenNebula :

onehost create onevmhost --im im_kvm --vm vmm_kvm --net dummy

oneadmin@onevmhost:~$ onehost list

ID	NAME	CLUSTER	RVM	TCPU	FCPU	ACPU	TMEM	FMEM	AMEM	STAT
0	onevmhost	-	2	800	796	600	7.5G	6.7G	5.5G	on

once you register a Host check the STAT flag. It should display "on".

You may need to debug log files if Value "Err" is display for STAT.

Note :Common cause of "Err" flag will be either password less connection to OneVMHost is lost or VMHost is not available to OneHost.

Hint: Just type onehost and press enter to get all available parameters.

Command "onehost top" will display the output of "onehost list" continuously.

In case of any errors just check ~/var/oned.log

Now let's create a cluster named "1cluster"
onecluster create 1cluster

Let's get the ID of 1cluster
oneadmin@onevmhost:/home/localadmin$ onecluster list

ID	NAME	HOSTS	VNETS	DATASTORES
100	1cluster	0	0	0

Add both the hosts to 1cluster
onecluster addhost 100 0

- To obtain detailed information about the registered host use the "show" function of "onehost" command

onehost show <host ID> /<host_Name>
e.g onehost show 0 or onehost show onevmhost

```
oneadmin@OneServer:~$ onehost show 0
HOST 0 INFORMATION
ID                : 0
NAME              : onevmhost
CLUSTER           : -
STATE             : MONITORING
IM_MAD            : im_kvm
VM_MAD            : vmm_kvm
VN_MAD            : dummy
LAST MONITORING TIME  : 1338727143

HOST SHARES
MAX MEM           : 8172324
USED MEM (REAL)   : 240480
USED MEM (ALLOCATED)  : 0
MAX CPU           : 800
USED CPU (REAL)   : 0
USED CPU (ALLOCATED)  : 0
MAX DISK          : 0
USED DISK (REAL)  : 0
USED DISK (ALLOCATED) : 0
RUNNING VMS       : 0

MONITORING INFORMATION
ARCH="x86_64"
CPUSPEED="1998"
FREECPU="800.0"
FREEMEMORY="7931844"
HOSTNAME="ONEVMHOST"
HYPERVISOR="kvm"
MODELNAME="Intel(R) Xeon(R) CPU E5440 @ 2.83GHz"
```

```
NETRX="0"
NETTX="0"
TOTALCPU="800"
TOTALMEMORY="8172324"
USEDCPU="0.0"
USEDMEMORY="240480"
oneadmin@OneServer:~$
```

Creating a Windows XP VM

Have a windows XP SP3 install disk as an ISO file and store it in /var/lib/image/iso folder

- e.g. /var/lib/image/iso/Win_XP_original.iso

Create a folder /var/lib/images and make oneadmin as the owner of the folder

```
mkdir /var/lib/image
chown -R oneadmin /var/lib/image
```

Create an empty Image file of 15 G

```
qemu-img create -f raw win-xp.img 15G
```

Create a Libvirt deployment file [/var/lib/image/win-deployment] and store below content.

nano /var/lib/image/win-deployment

```
<domain type='kvm' xmlns:qemu='http://libvirt.org/schemas/domain/qemu/1.0'>
<name>win-50</name>
<memory>1048576</memory>
<os>
<type arch='i686'>hvm</type>
<boot dev='hd'/>
<boot dev='cdrom'/>
</os>
<on_reboot>restart</on_reboot>
<on_crash>restart</on_crash>
<devices>
<emulator>/usr/bin/kvm</emulator>
<disk type='file' device='disk'>
<source file='/var/lib/image/win-xp.img'/>
<target dev='hda'/>
<driver name='qemu' type='raw' cache='default'/>
</disk>
<disk type='file' device='cdrom'>
<driver name='qemu' type='raw'/>
<target dev='hdc' bus='ide'/>
<readonly/>
<source file='/var/lib/image/iso/Win_XP_original.iso'/>
<address type='drive' controller='0' bus='1' unit='0'/>
</disk>
<controller type='ide' index='0'>
<address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x1'/>
</controller>
<!--use one network -->
<interface type='network'>
<source network='default'/>
</interface>
<graphics type='vnc' port='5950'/>
</devices>
<features>
<acpi/>
</features>
</domain>
```

Start virsh by typing “virsh” on the $prompt . You will be taken to Virtual-shell
On the virsh # prompt type the below and press enter

```
virsh # create /var/lib/image/win_deployment
```

Yor will get an output similar to below

virsh # create /var/lib/image/win_deployment

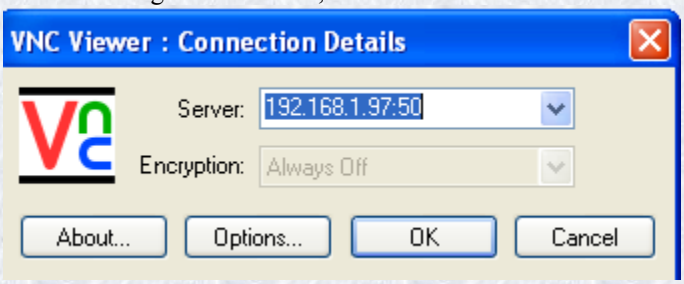
Domain win-50 created from /var/lib/image/win_deployment

You will be able to monitor the Windows Installation through a VNC console. To get the VNC console PORT # , type below commands [list and then vncdisplay] in Virsh# prompt

```
virsh # list
 Id Name                 State
----------------------------------
  9 win-50               running

virsh # vncdisplay 9
:50
```

Monitor Windows XP instillation through VNC viewer, with the IP → 192.168.1.97:50



When asked

- Provide password for Administrator.
- Just create only one user named "Admin"

Complete the Windows XP installation.

Once installation is completed , login to windowsXP and download Intel e1000 Lan driver . Just save it and donot install.
Download it from Intel site :http://downloadcenter.intel.com/detail_desc.aspx?agr=Y&DwnldID=18717
16) Filename: "PROWIN32.EXE"

Shutdown Windows
Open the file /var/lib/image/win-deployment in an editor and add a line<model type='e1000'/> to <interface> section

```
type='network'> section as highlighted below
<!--ONE Network-->
<interface type='network'>
<source network='default'/>
<model type='e1000'/>
</interface>
```

Start Windows by typing the below command in virsh # prompt. [same as we did earlier]

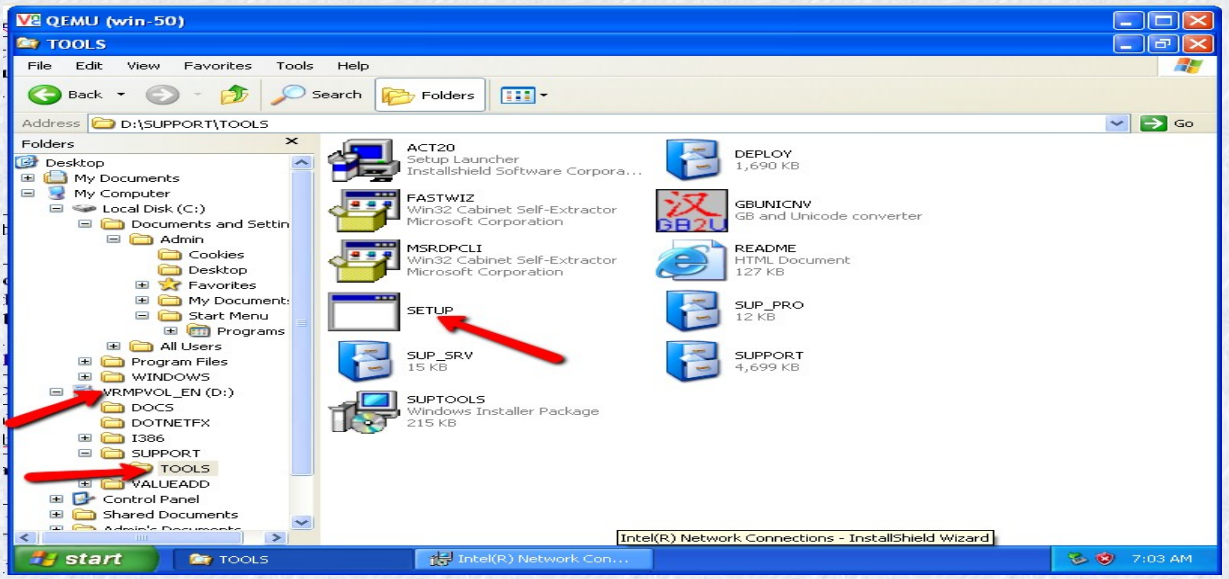
```
virsh# create /var/lib/image/win-deployment
```

Windows will detect the new N/w hardware and tries to install a driver. Let's not disturb the "Found new Hardware wizard" window. It will automatically disappear.
Double click PROWIN32.EXE and install Intel E1000 Network driver utility. Once completed you will have Intel e1000 Network driver installed.
[In the installation dialog, when asked Do not select the Checkbox for SNMP]

Enable remote desktop access [Control Panel-System-Remote tab]

Disable Windows Firewall -click the icon "Windows Security alerts" in the taskbar and switch firewall to OFF state in the resulting window.

Install Windows Support tools from the installation CDROM. We need it to run "netsh" command.
Go to folder D:\Support\Tools and Double click file "Setup.exe".
When asked Select " Complete" instead of "Typical" and complete the installation



Just to test it, open a command window and type

netsh int show interface

This will show the network interfaces available in the system.

Create two new folders in C drive , admin and autorun [within admin]

mkdir admin
cd admin
mkdir autorun
cd autorun

Create a StartupScript.bat in c:\admin\autorun folder, and store following content in it.

StartupScript.bat

```
:: Put whatever you want here. Just leave the bottom line untouched.
c:\Windows\System32\spoolsv.exe recycle apppool "DefaultAppPool"
:: Below file will be available only when WINDOWS instance starts as a guest OS in side VM. You just specify it for now.
call d:\setcontextvals.bat
```

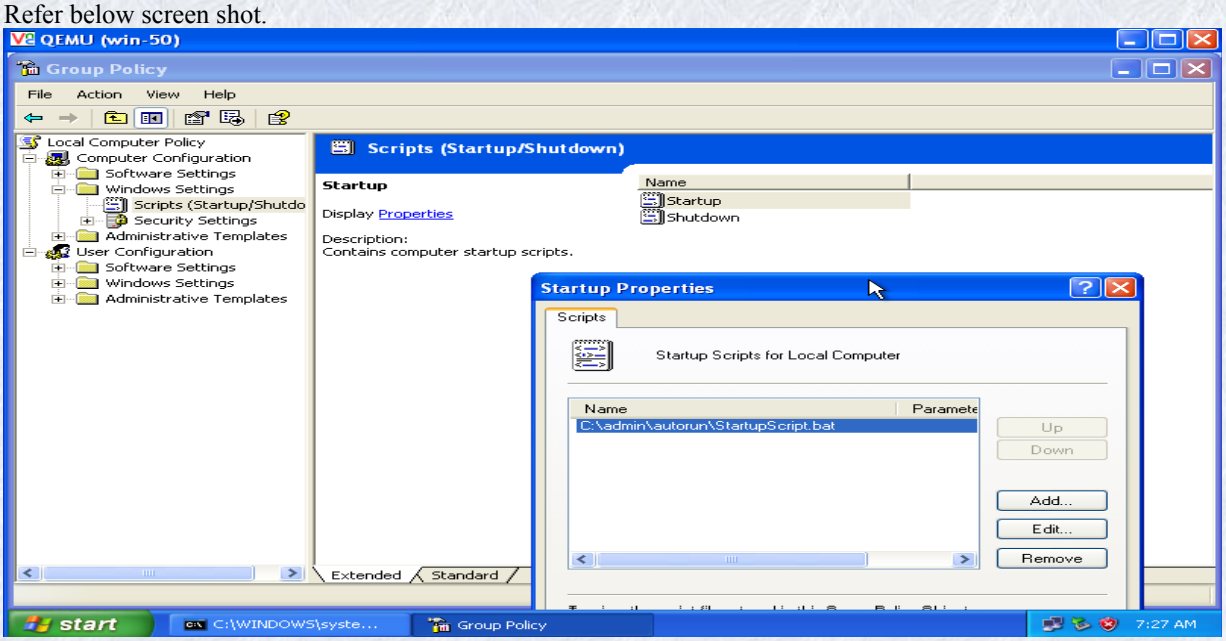
Currently you donot have d:\setcontextvals.bat file. But do not bother, we will soon create it with in OpenNebula front end and pass it on to the Windows instance.

Schedule StartupScript.bat to run on System startup:

2. Start –> Run –> "gpedit.msc"

3. Expand Computer Configuration\Windows Settings\Scripts
4. Double Click Startup
5. Click Add…
6. Enter c:\admin\autorun\ StartupScript.bat for the Script Name
7. Leave Script Parameters blank
8. Click the OK button

Refer below screen shot.



Shutdown Windows.

Now, convert the image file from "raw" to "qcow" so that one will have noissues in understanding it. Should be run from VMHost.

```
qemu-img convert -O qcow2 win-xp.img win-xp-2.qcow2
```

On successful completion, you will have a new file "win-xp-2.qcow2" created. The size will be much smaller and will be around 1.6 to 2G, depends on what you actually installed in Windows XP.

Make a copy of this file and preserve it somewhere.

Move the win-xp-2.qcow2 file to /var/lib/one/var/datastores/ folder

```
mv /var/lib/image/ win-xp-2.qcow2 /var/lib/one/var/datastores/ win-xp-2.qcow2
```

Let's now move to OpenNebula.

List the available datastores:

```
oneadmin@onevmhost:~$ onedatastore list
  ID NAME            CLUSTER  IMAGES TYPE  TM
   0 system                -       0 -     shared
   1 default               -       0 fs    shared
```

Create a folder "template" with in /var/lib/one to store all template files

```
mkdir /var/lib/one/template
```

Create a image definition template file /var/lib/one/template/Winxp.img and store below content in it. Save and Exit

```
NAME = "Winxp-NoPersistence"
SOURCE = /var/lib/one/var/datastores/win-xp-2.qcow2
```

TYPE = OS
PUBLIC = YES

Create an image with default datastore

```
oneimage create winxp.img  -d default
```

List the image to check the status of it. On successful creation It should show "Rdy" status.

```
oneadmin@onevmhost:~$ oneimage list
  ID USER     GROUP    NAME         DATASTORE     SIZE TYPE PER STAT  RVMS
   0 oneadmin oneadmin Winxp-NoPers default          0M  OS      No  rdy       0
oneadmin@onevmhost:~$ oneimage show 6
IMAGE 0 INFORMATION
ID            : 0
NAME          : Winxp-NoPersistence
USER          : oneadmin
GROUP         : oneadmin
DATASTORE     : default
TYPE          : OS
REGISTER TIME  : 06/03 15:47:18
PERSISTENT    : No
SOURCE        : /var/lib/one/var/datastores/win-xp.qcow2
SIZE          : 0
STATE         : used
RUNNING_VMS   : 0

PERMISSIONS
OWNER         : um-
GROUP         : ---
OTHER         : ---

IMAGE TEMPLATE
DEV_PREFIX="hd"
PUBLIC="YES"
```

In case of any errors during image creation, cross check the following,

1. Syntax errors in image creation template
2. Ownership of the qcow2 file should be with "oneadmin"

List the datastore again, you will see the value for "image" changes to "1" for default DS

```
oneadmin@onevmhost:~$ onedatastore list
  ID NAME            CLUSTER  IMAGES TYPE   TM
   0 system                         -      0     -    shared
   1 default                        -      1     fs   shared
```

Create a Vnet definition template /var/lib/one/template/winxp.net and store following content in it.Save exit

```
NAME = "private-win-d"
TYPE = RANGED
BRIDGE = br0
NETWORK_SIZE = C
IP_START = 192.168.1.151
IP_END = 192.168.1.254
VLAN = NO
NETWORK_MASK = 255.255.255.0
# Custom Attributes to be used in Context
GATEWAY = 192.168.1.1
DNS = 192.168.1.1
```

Create a Vnet,

```
onevnet create /var/lib/one/template/winxp.net
```

```
oneadmin@onevmhost:/home/localadmin$ onevnet list
  ID USER     GROUP    NAME          CLUSTER    TYPE BRIDGE  LEASES
   0 oneadmin oneadmin private-win-d  -          R    br0     0
```

Create a VM template file /var/lib/one/template/winxp.one and store following content in it. Check for the IMAGE ID and VnetID.

```
HOSTNAME = onevmhost
```

```
#CONTEXT definition section
CONTEXT=[FILES="/var/lib/one/.ssh/id_rsa.pub /var/lib/one/images/setcontextvals.bat /var/lib/one/images/sethostname.vbs",
HOSTNAME=WinXP-$VMID,
IP_PUBLIC="$NIC[IP, NETWORK=\"private-win-d\"]",
PASSWORD=redhat123,
ROOT_PUBKEY=id_rsa.pub,
USERNAME=Administrator]
#CAPACITY Definition
NAME=WindowsXP-NoPers
CPU=1
MEMORY=1024
# OS image, mapped to hda.
DISK=[ DRIVER=qcow2, READONLY=no, IMAGE_ID = 0, TARGET=hda, TYPE=disk ]
FEATURES=[ ACPI=yes ]
# I/O Devices Section
GRAPHICS=[ TYPE=vnc ]
#NETWORK Section:
NIC=[ model=e1000, network = "private-win-d" ]
#OS and BOOT Options Section
OS=[ ARCH=i686, BOOT=hd ]
#RAW Section
RAW=[ TYPE=kvm ]
```

Now let's add our VNET [ID 0] to the cluster
onecluster addvnet 100 0

Add the default datastore to the 1cluster
onecluster adddatastore 100 1

Now list the 1cluster again
oneadmin@onevmhost:/home/localadmin$ onecluster list

```
  ID NAME            HOSTS VNETS DATASTORES
 100 1cluster            1     1          1
```

List the VMs running currently
oneadmin@onevmhost:/home/localadmin$ onevm list

```
   ID USER     GROUP    NAME        STAT      CPU     MEM        HOSTNAME        TIME
```

None:

Store/preserve the VM template in OpenNebula using onetemplate command
onetemplate create ~/template/winxp.one

Test the creation using List and show commands

```
oneadmin@onevmhost:~$ onetemplate list
  ID USER     GROUP    NAME                    REGTIME
   0 oneadmin oneadmin WindowsXP-NoPer      05/28 09:50:44
oneadmin@onevmhost:~$ onetemplate show 0
TEMPLATE 6 INFORMATION
ID          :0
NAME          : WindowsXP-NoPers
USER          : oneadmin
GROUP         : oneadmin
REGISTER TIME  : 05/28 09:50:44

PERMISSIONS
OWNER         : um-
GROUP         : ---
OTHER         : ---

TEMPLATE CONTENTS
CONTEXT=[
  FILES="/var/lib/one/.ssh/id_rsa.pub /var/lib/one/images/setcontextvals.bat /var/lib/one/images/sethostname.vbs",
  HOSTNAME="WinXP-$VMID",
  IP_PUBLIC="$NIC[IP, NETWORK=\"private-win-d\"]",
  PASSWORD="redhat123",
  ROOT_PUBKEY="id_rsa.pub",
  USERNAME="Administrator" ]
CPU="1"
DISK=[
  DRIVER="qcow2",
```

```
  IMAGE_ID="0",
  READONLY="no",
  TARGET="hda",
  TYPE="disk" ]
FEATURES=[
  ACPI="yes" ]
GRAPHICS=[
  TYPE="vnc" ]
HOSTNAME="onevmhost"
MEMORY="1024"
NAME="WindowsXP-NoPers"
NIC=[
  MODEL="e1000",
  NETWORK="private-win-d" ]
OS=[
  ARCH="i686",
  BOOT="hd" ]
RAW=[
  TYPE="kvm" ]
TEMPLATE_ID="0"
```

Create the files specified in the CONTEXT section

- Create a folder /var/lib/one/images
  mkdir /var/lib/one/images
- Create a file /var/lib/one/images/setcontextvals.bat and store below content in it . This batch script will extract the IP address and HOSTNAME from the context.sh file. Context.sh file will be automatically created by OpenNebula based on the CONTEXT section in the VM Template file, VM Creation process, stores the context.sh file in a Virtual CDROM (for Windows instance , assume the CDROM is "D" drive) with the VM instance.

```
@echo off
setlocal ENABLEDELAYEDEXPANSION
for /F "skip=3 eol= tokens=*" %%S in (d:\context.sh) do (set line2=%%S&goto:forend2)
:forend2
REM echo line2=%line2%
set IPADD=%line2:~11,13%
REM echo %IPADD%
for /F "skip=2 eol= tokens=*" %%S in (d:\context.sh) do (set line1=%%S&goto:forend3)
:forend3
REM echo line1=%line1%
rem set hostname=!line1:~10,8!
rem echo hostname is !hostname!

setlocal
set str=%line1:HOSTNAME=%%
set str=%str:, =%
rem set hostname=%str:~1,-1%
REM echo %hostname%
call :dequote  %str%
REM echo ret=%ret%
endlocal
goto :eof

:dequote
setlocal
rem The tilde in the next line is the really important bit.
set thestring=%~1
endlocal&set ret=%thestring%
call d:\sethostname.vbs %ret%
netsh int ip set address "Local Area Connection 2" static %IPADD% 255.255.255.0 192.168.1.1 1
netsh int ip set dns "Local Area Connection 2" static 192.168.1.1 primary

goto :eof
```

- Create a file /var/lib/one/images/sethostname.vbs. Save and exit.
- This VB Script will change the hostname to the value passed by the CONTEXT section. [e.g. WinXp-39]

```
nHOSTNAME = Wscript.Arguments(0)
```

```
sNewName = nHOSTNAME
Set oShell = CreateObject ("WSCript.shell")
sCCS = "HKLM\SYSTEM\CurrentControlSet\"
sTcpipParamsRegPath = sCCS & "Services\Tcpip\Parameters\"
nHOSTNAME = Wscript.Arguments(0)
sNewName = nHOSTNAME
Set oShell = CreateObject ("WSCript.shell")
sCCS = "HKLM\SYSTEM\CurrentControlSet\"
sTcpipParamsRegPath = sCCS & "Services\Tcpip\Parameters\"
sCompNameRegPath = sCCS & "Control\ComputerName\"
With oShell
.RegDelete sTcpipParamsRegPath & "Hostname"
.RegDelete sTcpipParamsRegPath & "NV Hostname"
.RegWrite sCompNameRegPath & "ComputerName\ComputerName", sNewName
.RegWrite sCompNameRegPath & "ActiveComputerName\ComputerName", sNewName
.RegWrite sTcpipParamsRegPath & "Hostname", sNewName
.RegWrite sTcpipParamsRegPath & "NV Hostname", sNewName
End With ' oShell
rem Dim objShell
rem Set objShell = WScript.CreateObject("WScript.Shell")
rem objShell.Run "C:\WINDOWS\system32\shutdown.exe -r -t 0"
```

Instantiate the VM using onetemplate command

```
onetemplate instantiate 0
```

Monitor the progress of VM creation using onevm top command

```
onevm top
  ID USER     GROUP    NAME        STAT CPU     MEM        HOSTNAME        TIME
   2 oneadmin oneadmin one-2       runn   0        0K          onevmhost       0d 00:07
```

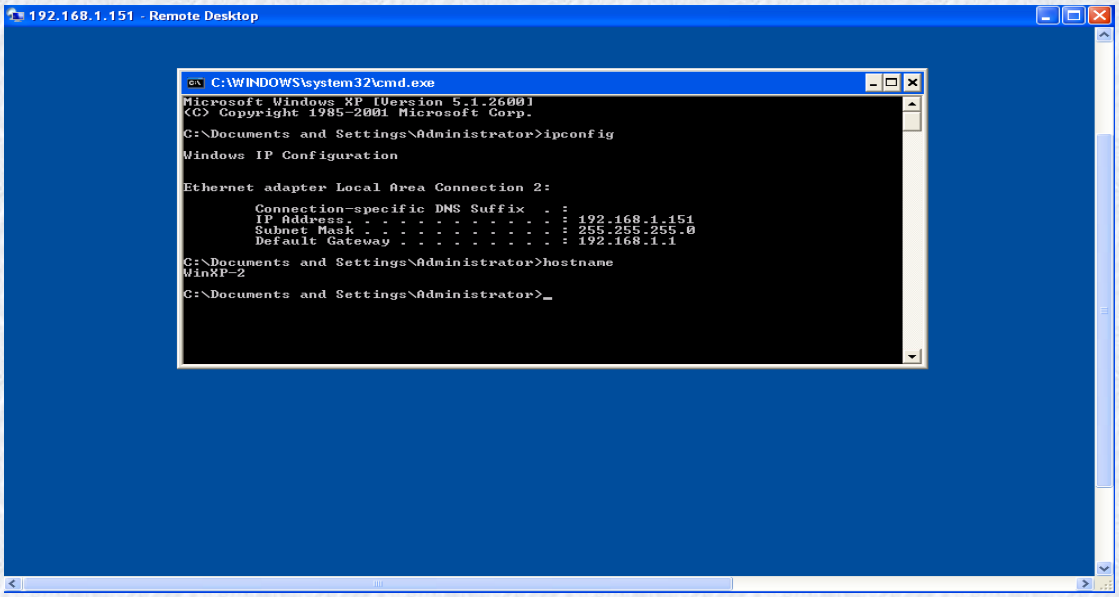
Note:I already had 2 Vms created in between. Hence the number is 2

Get the IP address and PORT of the Running VM

```
oneadmin@onevmhost:~$ onevm show 2 | grep IP
  IP_PUBLIC="192.168.1.151",
  IP="192.168.1.151",
oneadmin@onevmhost:~$ onevm show 2 | grep PORT
  PORT="5902",
```

Test by pinging the IP 192.168.1.151. It will Ping. If it does not Ping, Check through VNC console that whether you have disabled the Windows Firewall. Disable it.

Connect to Windows instance through Remote desktop.
Check to see if the IP address is set to 192.168.1.151 and hostname is changed to WinXP-2.
That's it. You have successfully created a Windows VM and set the IP address and hostname as passed by OpenNebula CONTEXT section.



Now lets continue with Heatbeat / High Availability

Testing "High Availability"

Let's create a new VM. Windows image file being quite big 2+ GB it will take some time during the prolog session. While PROLOG is on we will induce a FAILURE STATE , by stopping the heartbeat in Server1, the PROLOG process should continue with no hindrance from Server 2. Also the currently running VM instance should continue normally

Let's first check from which one is the Active Server now, using "df -h" command

root@server1:~# df -h
Filesystem Size Used Avail Use% Mounted on
/dev/sda5 4.6G 1.2G 3.3G 27% /
udev 238M 4.0K 238M 1% /dev
tmpfs 99M 252K 99M 1% /run
none 5.0M 4.0K 5.0M 1% /run/lock
none 246M 0 246M 0% /run/shm
/dev/sda1 92M 30M 57M 35% /boot
/dev/drbd0 64G 8.2G 52G 14% /var/lib/one

root@server2:~# df -h
Filesystem Size Used Avail Use% Mounted on
/dev/sda5 4.6G 1.1G 3.3G 25% /
udev 238M 4.0K 238M 1% /dev
tmpfs 99M 252K 99M 1% /run
none 5.0M 4.0K 5.0M 1% /run/lock
none 246M 0 246M 0% /run/shm
/dev/sda1 92M 30M 57M 35% /boot

So, Server 1 is active now.

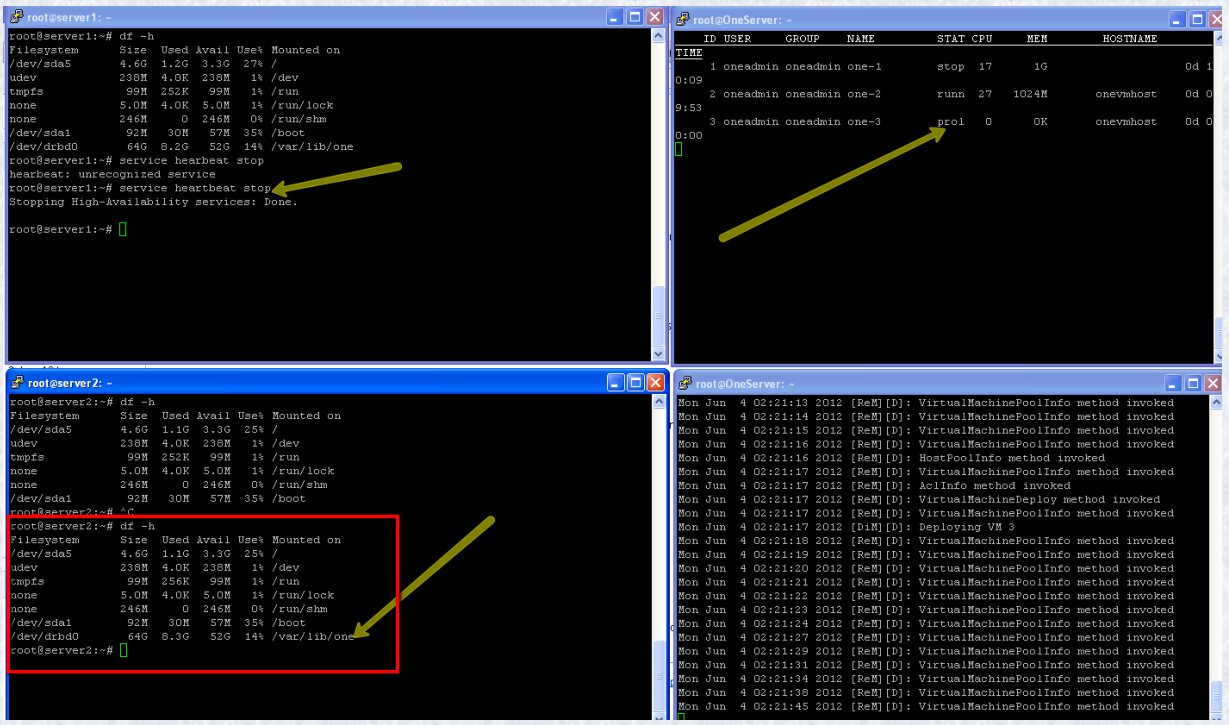
Create a new VM

onetemplate instantiate 0

Once VM enters "prolog" state, just stop heartbeat in Server 1.
root@server1:~# service heartbeat stop
Stopping High-Availability services: Done.

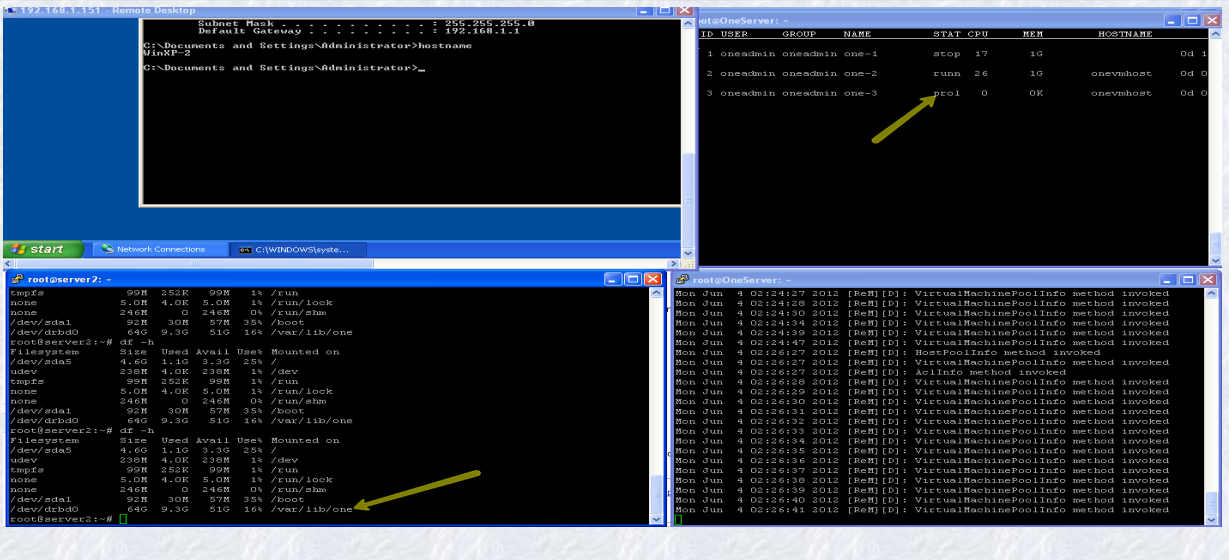
root@server2:~# df -h

```
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda5       4.6G  1.1G  3.3G  25% /
udev            238M  4.0K  238M   1% /dev
tmpfs            99M  252K   99M   1% /run
none            5.0M  4.0K  5.0M   1% /run/lock
none            246M     0  246M   0% /run/shm
/dev/sda1        92M   30M   57M  35% /boot
/dev/drbd0       64G   10G   50G  17% /var/lib/one
```

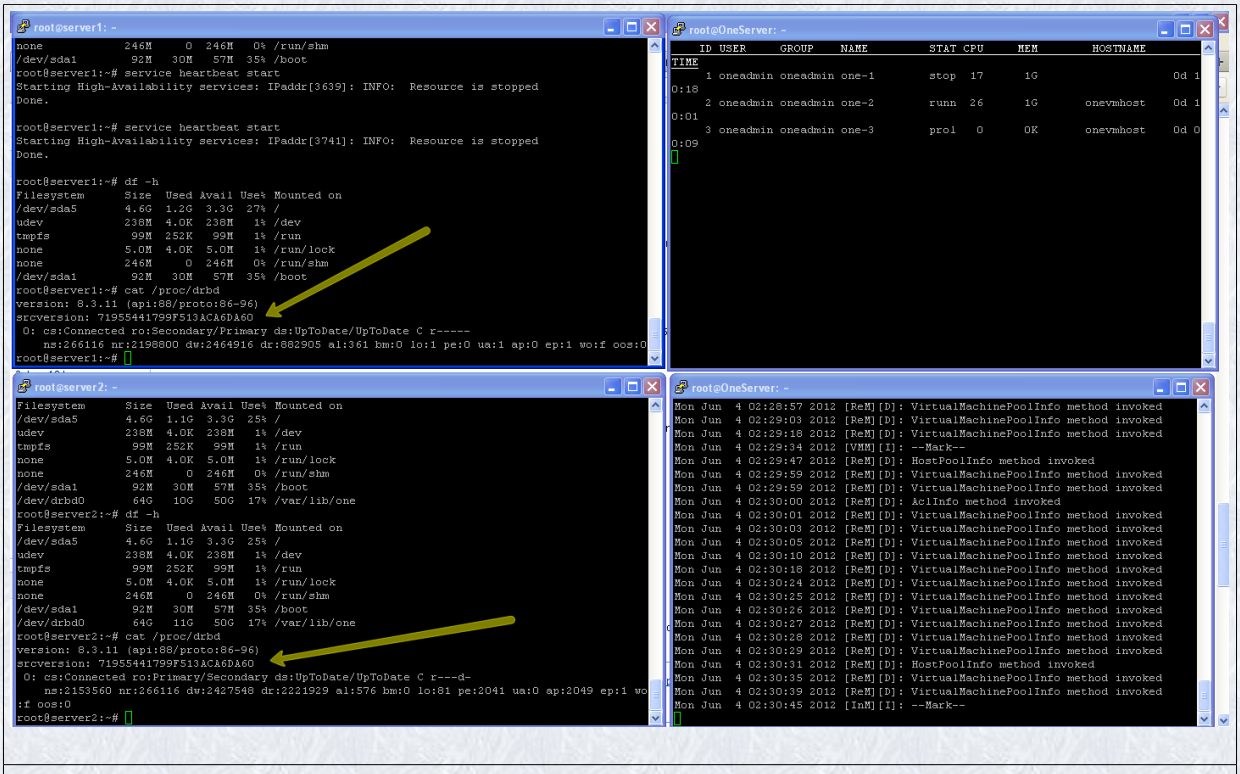


Once we stopped the heatbeat in Server1 , Server 2 became Active instantly and PROLOG state continued.

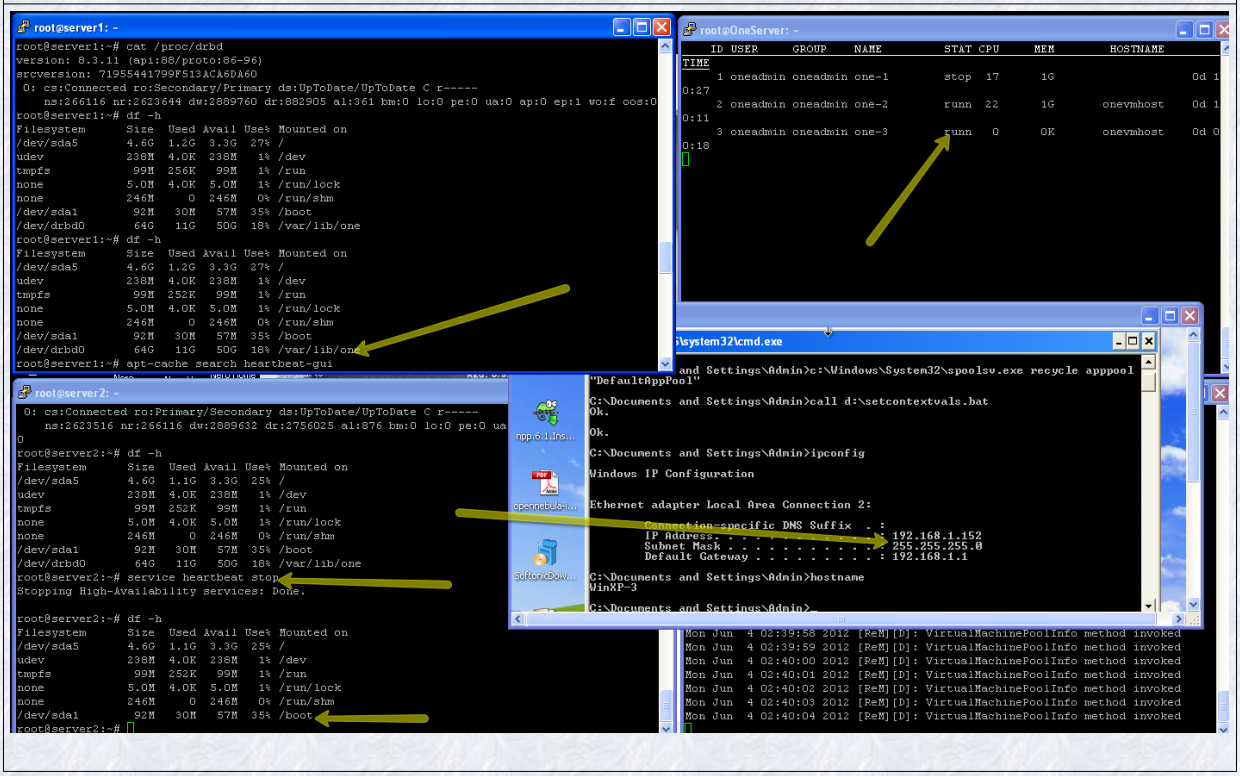
The existing VM is also running fine. Log file is running fine....



Check if DRBD also changes the PRIMARY and SECONDARY accordingly



After some time, let's reverse the proccess, now stop heartbeat in Server 2 and se what happens. Server 1 became Active and VM-3 also entered in to running state. We are able to connect to the Windows instance.



That's it.

If you liked this tutorial just put a comment to cloud.b.lab@zoho.com