Installation of Xen, XAPI (XenAPI) and Openstack on Ubuntu 13.04

XCP (Xen Cloud Platform) is the open source version similar to Citrix XenServer that uses the Xen Hypervisor. It is currently distributed as an ISO installer also called as XCP appliance. XCP uses XAPI or XenAPI to manage Xen hosts. XCP is based on CentOS 5.5

Project Kronos is an initiative to port the XAPI tool stack to Debian and Ubuntu. It is a management stack implemented in OCaml that configures and controls Xen hosts, attached storage, networking and virtual machine life cycle. It exposes a HTTP API and provides a command line interface (xe) for resource management.

Prerequisites

- Ubuntu 13.04 server
- Small root file system partition and have a large space dedicated for a LVM volume.
- root access to the host

Installing and configuring XAPI (XenAPI)

1. Install XCP-XAPI
    - # apt-get install xcp-xapi

      – choose bridge when prompted for network backend

2. Setup GRUB to boot the Xen Hypervisor
    - # sed -i ‘s/GRUB_DEFAULT=.*+/GRUB_DEFAULT=”Xen 4.2-amd64″/’ /etc/default/grub

3. Disable apparmor at boot
    - # sed -i ‘s/GRUB_CMDLINE_LINUX=.*+/GRUB_CMDLINE_LINUX=”apparmor=0″/’ /etc/default/grub

4. Restrict dom0 to 2GB of memory and 2 vcps
    - # vi /etc/default/grub

      after GRUB_CMDLINE_LINUX=”apparmor=0″ add line

    - GRUB_CMDLINE_XEN=”dom0_mem=2G,max:2G dom0_max_vcpus=2″

5. Update Grub with the config changes we just made

   # update-grub

6. Once the server is back online ensure that Xen is running
    - cat /proc/xen/capabilities should display “control_d”

if 'cat /proc/xen/capabilities' doesn't return anything, add the following line to /etc/fstab :

```
none    /proc/xen      xenfs   defaults       0       0
```

And then do mount -a.

7. Setup the default toolstack
   - `# vi /etc/default/xen`

     – set ‘TOOLSTACK=xapi’

8. Disable xend from starting at boot
   - `# sed -i -e ‘s/xend_start$/#xend_start/’ -e ‘s/xend_stop$/#xend_stop/’ /etc/init.d/xend`

     NOTE: only xend the deamon needs to be disabled from starting, /etc/init.d/xend handles other things like modules and xenfs. Do not disable it from the runlevel

9. Disable service xendomains
   - `# update-rc.d xendomains disable`

10. Fix for qemu which emulates the console does not have the keymaps in the correct location
    - `# mkdir /usr/share/qemu;`
    - `# ln -s /usr/share/qemu-linaro/keymaps /usr/share/qemu/keymaps`

11. check ifconfig command output
    - If the ethernet device is recognized as em1 instead of eth0.

Then change it by adding 'biosdevname=0' to GRUB_CMDLINE_LINUX in /etc/default/grub.
then hit update-grub command and reboot machine.

12. Setup bridge networking
    - `# vi /etc/network/interfaces`

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).
# The loopback network interface
auto lo
iface lo inet loopback# The primary network interface
auto xenbr0
iface xenbr0 inet static
address < eth0 ip address here >
netmask < eth0 netmask address here >
network < eth0 network address here >
broadcast < eth0 broadcast address here >
gateway
# dns-* options are implemented by the resolvconf package, if installed
dns-nameservers 4.2.2.2
bridge_ports eth0
iface eth0 inet manual
```

13. Configure xcp to use bridge networking instead of openswitch
    - `# vi /etc/xcp/network.conf`
      replace “openswitch” with “bridge”

14. All set – ready to reboot and let xcp-xapi toolstack take over
    - `# reboot`

15. On restart – confirm that xcp is working

    -
      ```
      # xe vm-list
      uuid ( RO)          : 92ddb581-e6a8-2e6d-045e-d35b22f01668
      name-label ( RW): Control domain on host: ramanujan
      power-state ( RO): running
      ```

    - If your output looks similar – xapi is running on the server, if you get “Connection refused” then xapi is not setup correctly

Setting up the LVM storage volume

Assuming that you configured a large partition for use as a LVM volume during installation, this part sets up the partition and adds it as a local storage repository. By default the volume group, and a logical volume will be created if

1. verify that you have a LVM partition
    - `# fdisk -l`
    - This should list a partition of type “Linux LVM”. If you don’t see a partition and you have free space on the disk, create a new partition of type “Linux LVM” (8e)
    - If you have partition of type “Linux LVM” follow the steps below

2. create a physical volumes
    - `# pvcreate /dev/cciss/c0d0p2`
    - `# pvdisplay`

      You should see similar output

      ```
      “/dev/cciss/c0d0p2″ is a new physical volume of “947.60 GiB”
      — NEW Physical volume —
      PV Name /dev/cciss/c0d0p2
      VG Name
      PV Size 947.60 GiB
      Allocatable NO
      PE Size 0
      Total PE 0
      Free PE 0
      Allocated PE 0
      PV UUID rNeGnf-TbJS-vfSm-t7la-wNCv-Lpc3-vjn33c
      ```

3. create a volume group
    - `# vgcreate VolumeGroup /dev/cciss/c0d0p2`

- # pvdisplay - this should display the volume group we created on the physical volume

  ```
  — Physical volume —
  PV Name /dev/cciss/c0d0p2
  VG Name VolumeGroup
  PV Size 947.60 GiB / not usable 2.90 MiB
  Allocatable yes
  PE Size 4.00 MiB
  Total PE 242584
  Free PE 242584
  Allocated PE 0
  PV UUID rNeGnf-TbJS-vfSm-t7la-wNCv-Lpc3-vjn33c
  ```

4. create a logical volume on “VolumeGroup”
   - # lvcreate –size 947G -n LocalStorage VolumeGroup
   - # lvdisplay

     – this should display the logical volume we created on the volume group

     ```
     — Logical volume —
     LV Name /dev/VolumeGroup/LocalStorage
     VG Name VolumeGroup
     LV UUID pCWgAs-cpfh-IAdU-uVMi-EJbo-iy2x-TlMzar
     LV Write Access read/write
     LV Status available
     # open 0
     LV Size 947.00 GiB
     Current LE 242432
     Segments 1
     Allocation inherit
     Read ahead sectors auto
     - currently set to 256
     Block device 252:0
     ```

5. Register the logical volume for use with XAPI
   - xe sr-create type=ext name-label=Local Storage device-config:device=/dev/mapper/VolumeGroup-LocalStorage

     – this will take a while if the volume is large

   - # xe sr-list name-label="Local Storage"

     – this should display the storage repository

     ```
     uuid ( RO) : 7dea0028-ee94-6c16-2f61-c699ed4a1d18
     name-label ( RW): Local Storage
     name-description ( RW):
     host ( RO): ubuntu-xenserver-1
     type ( RO): ext
     content-type ( RO):
     ```

- # xe pool-param-set uuid=<pool-uuid> default-SR=<sr-uuid>
  Get the pool-uuid from- xe pool-list
  and get sr-uuid from - xe sr-list

Setup openstack

Prerequisite:

```
# apt-get install apache2
# apt-get install unzip
# xe template-list
```

This should return a list of templates. If it returns nothing, then run-

```
# /usr/lib/xcp/lib/create_templates
```

Delete template 'jeos_template_for_devstack' if exists.

- Get template uuid for - jeos_template_for_devstack
  ```
  # xe template-list name-label= jeos_template_for_devstack
  ```
- Uninstall the template
  ```
  # xe template-uninstall template-uuid=<uuid of template>
  ```

Delete template 'Ubuntu 13.04 (64-bit) for DevStack' if exists.

- Get template uuid for - Ubuntu 13.04 (64-bit) for DevStack
  ```
  # xe template-list name-label="Ubuntu 13.04 (64-bit) for DevStack"
  # xe template-param-set uuid=<template-uuid> other-config:default_template=false
  # xe template-param-set is-a-template=false uuid=<template-uuid>
  # xe vm-uninstall uuid=<template-uuid>
  ```

Step 1 : Download devstack on dom0

```
# git clone git://github.com/openstack-dev/devstack.git
# cd devstack
```

Step 2: Configure your localrc inside the devstack directory

Devstack uses a localrc for user-specific configuration. Note that the XENAPI_PASSWORD must be your dom0 root password. Of course, use real passwords if this machine is exposed.

```
cat > ./localrc <<EOF
# Passwords
# NOTE: these need to be specified, otherwise devstack will try
# to prompt for these passwords, blocking the install process.

MYSQL_PASSWORD=my_super_secret
SERVICE_TOKEN=my_super_secret
ADMIN_PASSWORD=my_super_secret
SERVICE_PASSWORD=my_super_secret
RABBIT_PASSWORD=my_super_secret
SWIFT_HASH="66a3d6b56c1f479c8b4e70ab5c2000f5"
# This will be the password for the OpenStack VM (both stack and root users)
GUEST_PASSWORD=my_super_secret

# XenAPI parameters
```

```
# NOTE: The following must be set to your XenServer root password!

XENAPI_PASSWORD=my_dom0_root_password
XENAPI_CONNECTION_URL="http://address_of_your_dom0"
VNCSERVER_PROXYCLIENT_ADDRESS=address_of_your_dom0

# Download a vhd and a uec image
IMAGE_URLS="\
https://github.com/downloads/citrix-openstack/warehouse/cirros-0.3.0-x86_64-disk.vhd.tgz,\
http://download.cirros-cloud.net/0.3.1/cirros-0.3.1-x86_64-uec.tar.gz"

# Explicitly set virt driver
VIRT_DRIVER=xenserver

# Explicitly enable multi-host for nova-network HA
MULTI_HOST=1

# Give extra time for boot
ACTIVE_TIMEOUT=45

EOF
```

Update file : /root/devstack/tools/xen/xenrc with-

```
UBUNTU_INST_RELEASE="raring"
UBUNTU_INST_TEMPLATE_NAME="Ubuntu 13.04 (64-bit) for DevStack"
```

Step 3: Run ./install_os_domU.sh from the tools/xen directory

```
# cd tools/xen
#  ./install_os_domU.sh
```

Once this script finishes executing, log into the VM (openstack domU) that it installed and tail the run.sh.log file. You will need to wait until it run.sh has finished executing.

Create a directory in dom0

```
# mkdir -p /usr/etc/
```

Step 4: To access openstack dashboard

http://IP-of-new-openstack-domu

Step 5: To create new VM from openstack

1) Login

Username : stack

Password : get password from file : /root/devstack/localrc on dom0 , search for GUEST_PASSWORD=**my_super_secret**

2) Download ubuntu-server-cloud image

```
# cd /opt/stack/devstack/files
# wget https://cloud-images.ubuntu.com/releases/raring/release/ubuntu-13.04-server-cloudimg-amd64.tar.gz
```

3) Extract image

```
# cd /opt/stack/devstack/files/images
# mkdir ubuntu-raring
# cd  ubuntu-raring
# tar -xzvf /opt/stack/devstack/files/raring-server-cloudimg-amd64.tar.gz
# source /opt/stack/devstack/openrc
```

4) Add the image to glance:

```
# glance image-create --name=ubuntu-raring-image --is-public=true --container-format=ami --disk-format=ami < raring-server-cloudimg-amd64.img
```

Check that adding the image was successful (Status should be ACTIVE when the operation is complete):

```
# glance image-list
```

5) Create a keypair so you can ssh to the instance:

```
# nova keypair-add raring > raring.pem
# chmod 600 raring.pem
```

6) Run (boot) a test instance:

```
# nova boot --image ubuntu-raring-image --flavor m1.small --key_name raring my-ubuntu-server
```

Here's a description of the parameters used above:

--image: the name or ID of the image we want to launch, as shown in the output of nova image-list

--flavor: the name or ID of the size of the instance to create (number of vcpus, available RAM, available storage). View the list of available flavors by running nova flavor-list

-key_name: the name of the key to inject into the instance at launch.

7) Check the status of the instance you launched:

```
# nova list
 After instance become ACTIVE, connect to it using :

 ipaddress=... # Get IP address from "nova list"
 ssh -i raring.pem -l ubuntu #ipaddress
```