Setting up VirtualBox Environment

1. Create following Networks in VirtualBox:
    # Host-Only Ethernet Adapter #2, Configure Manually, IPv4 Address: 10.0.0.1, IPv4 Network Mask: 255.255.255.0, no DHCP
    # NAT Network “ProviderNetwork1”, Network CIDR: 203.0.113.0/24
2. Create following nodes in VirtualBox:
    # Name: “controller”
        i. 2 VCPU
        ii. 6144 MB RAM
        iii. 20GB Primary Disk
        iv. Network Interfaces:
            - Adapter 1: connected to NAT
            - Adapter 2: connected to Host-Only Network #2
            - Adapter 3: connected to NAT Network “ProviderNetwork”, enable Promiscuous Mode
    # Name: “compute1”
        i. 1 VCPU
        ii. 2048 MB RAM
        iii. 10GB Primary Disk
        iv. Network Interfaces:
            - Adapter 1: connected to NAT
            - Adapter 2: connected to Host-Only Network #2
            - Adapter 3: connected to NAT Network “ProviderNetwork”, enable Promiscuous Mode
    # Name: “compute2”
        i. 1 VCPU
        ii. 2048 MB RAM
        iii. 10GB Primary Disk
        iv. Network Interfaces:
            - Adapter 1: connected to NAT
            - Adapter 2: connected to Host-Only Network #2
            - Adapter 3: connected to NAT Network “ProviderNetwork”, enable Promiscuous Mode
    # Name: “block1”
        i. 1 VCPU
        ii. 1024 MB RAM
        iii. 10GB Primary Disk + Second Disk 30GB
        iv. Network Interfaces:
            - Adapter 1: connected to NAT
            - Adapter 2: connected to Host-Only Network #2
            - Adapter 3: connected to NAT Network “ProviderNetwork”, enable Promiscuous Mode
    # Name: “deployment”
        i. 1 VCPU
        ii. 2048 MB RAM

iii. 40GB Disk
iv. Network Interfaces:
- Adapter 1: connected to NAT
- Adapter 2: connected to Host-Only Network #2

Ubuntu Linux Installation on Target VMs

3. Install Ubuntu 16.04 Server in Minimal VM configuration on each node.

Preparing Target VMs

4. On Each Node, configure GRUB to enable standard Network Interface Names:
   # Edit /etc/default/grub and modify the GRUB_CMDLINE_LINUX to "net.ifnames=0 biosdevname=0"
   # Run "update-grub" as superuser
   # Run "visudo" to enable primary user to "sudo su" without password:
   ```
   <username> ALL=(ALL) NOPASSWD:ALL
   ```
   # Set up /etc/hosts:
   ```
   127.0.0.1  localhost
   10.0.0.11  controller
   10.0.0.31  compute1
   10.0.0.32  compute2
   10.0.0.41  block1
   10.0.0.100 deployment
   ```
5. On **controller**, **compute1** and **compute 2** set up /etc/network/interfaces:
```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp

auto eth1
iface eth1 inet static
  address <node ip address>
  netmask 255.255.255.0
  dns-nameservers 8.8.8.8

auto eth2
iface eth2 inet manual
  up ip link set dev eth2 up
  down ip link set dev eth2 down
```
6. On **controller** run following commands as superuser:
```
  #  apt update -y
  #  apt upgrade -y
  #  apt install -y python python-simplejson glances vim
  #  reboot
```

7. On **compute1** and **compute2** run following commands as superuser:

```
#  apt update -y
#  apt upgrade -y
#  apt install python python-simplejson glances vim
#  echo "configfs" >> /etc/modules
#  update-initramfs -u
#  reboot
```

8. On **block1** set up /etc/network/interfaces:

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp

auto eth1
iface eth1 inet static
  address 10.0.0.41
  netmask 255.255.255.0
  dns-nameservers 8.8.8.8
```

9. On **block1** run following commands as superuser:

```
#  apt update -y
#  apt upgrade -y
#  apt install -y python python-simplejson glances vim
#  apt install -y lvm2 thin-provisioning-tools
#  pvcreate /dev/sdb
#  vgcreate cinder-volumes /dev/sdb
#  echo "configfs" >> /etc/modules
#  update-initramfs -u
#  reboot
```

Prepare deployment VM

1. Configure GRUB to enable standard Network Interface Names:
   - # Edit /etc/default/grub and modify the GRUB_CMDLINE_LINUX to "net.ifnames=0 biosdevname=0"
   - # Run "update-grub" as superuser
   - # Run "visudo" to enable primary user to "sudo su" without password:
     `<username> ALL=(ALL) NOPASSWD:ALL`
   - # Set up /etc/hosts:

```
127.0.0.1  localhost
10.0.0.11  controller
10.0.0.31  compute1
10.0.0.32  compute2
10.0.0.41  block1
10.0.0.100 deployment
```

2. Set up /etc/network/interfaces:

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp

auto eth1
iface eth1 inet static
  address 10.0.0.100
  netmask 255.255.255.0
  dns-nameservers 8.8.8.8
```

3. Run following commands as superuser:

```
#  apt update -y
#  apt install -y python-jinja2 python-pip libssl-dev curl
   glances vim
#  pip install -U pip
#  apt upgrade -y
#  reboot
```

4. Configure & test **ssh** access to other nodes:

```
#  ssh-keygen -t rsa
#  ssh-copy-id <username>@controller
#  ssh-copy-id <username>@compute1
#  ssh-copy-id <username>@compute2
#  ssh-copy-id <username>@block1
#  ssh <username>@controller
#  ssh <username>@compute1
#  ssh <username>@compute2
#  ssh <username>@block1
```

5. Verify network configuration:

```
#  ifconfig
#  ip a
```

6. Install Ansible and Kolla-Ansible:

```
#  pip install ansible==2.5.2
#  pip install kolla-ansible==6.0.0
#  cp -r /usr/local/share/kolla-ansible/etc_examples/kolla
   /etc/kolla
```

7. Prepare **nova-compute.conf** file with QEMU configuration:

```
#  mkdir -p /etc/kolla/config/nova
#  vim /etc/kolla/config/nova/nova-compute.conf
   [libvirt]
   virt_type = qemu
   cpu_mode = none
```

Kolla-Ansible Parameters

1. Edit /etc/kolla/**globals.yml** file:
   config_strategy: “COPY_ALWAYS”
   kolla_base_distro: “centos”
   kolla_install_type: “binary”
   kolla_release: “pike”
   kolla_internal_vip_address: “10.0.0.10”
   network_interface: “eth1”
   neutron_external_interface: “eth2”
   neutron_plugin_agent: “openvswitch”
   keepalived_virtual_router_id: “51”
   nova_console: “novnc”
   enable_cinder: “yes”
   enable_cinder_backup: “no”
   enable_cinder_backend_iscsi: “yes”
   enable_cinder_backend_lvm: “yes”
   enable_ha_proxy: “yes”
   enable_heat: “yes”
   enable_horizon: “yes”
   enable_openvswitch: “{{ neutron_plugin_agent != ‘linuxbridge’ }}”
   keystone_token_provider: “fernet”
   fernet_token_expiry: 86400
   glance_backend_file: “yes”
   cinder_volume_group: “cinder-volumes”
   nova_compute_virt_type: “qemu”
2. Edit multimode file:

```
#  cp /usr/local/share/kolla-ansible/ansible/inventory/* .
#  vim multimode
   [control]
   controller ansible_ssh_user=<username>
   ansible_become=True
   ansible_private_key_file=/root/.ssh/id_rsa
   [network]
   controller ansible_ssh_user=<username>
   ansible_become=True
   ansible_private_key_file=/root/.ssh/id_rsa
   [compute]
   compute1          ansible_ssh_user=<username>
   ansible_become=True
   ansible_private_key_file=/root/.ssh/id_rsa
   compute2          ansible_ssh_user=<username>
   ansible_become=True
   ansible_private_key_file=/root/.ssh/id_rsa
   [monitoring]
```

```
controller  ansible_ssh_user=<username>
ansible_become=True
ansible_private_key_file=/root/.ssh/id_rsa
[storage]
block1            ansible_ssh_user=<username>
ansible_become=True
ansible_private_key_file=/root/.ssh/id_rsa
```

Running Kolla-Ansible Deployment

1. Generate Kolla-Ansible passwords:
   ```
   #  kolla-genpwd
   ```
2. Bootstrap Nodes:
   ```
   #  kolla-ansible -i multinode bootstrap-servers
   ```
3. Run pre-deployment checks:
   ```
   #  kolla-ansible -i multinode prechecks
   ```
4. Run Deployment:
   ```
   #  kolla-ansible -i multinode deploy
   ```
5. Verify that containers are up, not restarting:
   ```
   #  docker ps -a
   ```

Finalize Deployment

1. Run Kolla Post Deploy script:
   ```
   #  kolla-ansible post-deploy
   ```
2. Install OpenStack Client:
   ```
   #  pip install python-openstackclient
   ```
3. Populate OpenStack with initial set up:
   ```
   #  vim /usr/local/share/kolla-ansible/init-runonce
   #  . /etc/kolla/admin-openrc.sh
   #  cd /usr/local/share/kolla-ansible
   #  ./init-runonce
   ```

Verification and Using the OpenStack

1. Check the **admin** password:
   ```
   #  grep keystone_admin_password /etc/kolla/passwords.yml
   ```
2. Open browser and go to 10.0.0.10, login as admin