OpenStack Project Part 3 – Orchestration

For this part of the OpenStack Project, we will be prepping our orchestration tool (ansible, to run the playbooks developed by the creators of the Kolla-Ansible project) and doing some more work with docker. Our OpenStack cloud will be comprised of services configured to run as docker containers. Each OpenStack service that is configured for us (via Ansible) will run as a separate docker container.

We will specifically be deploying **OpenStack Ussuri** (keep this in mind when searching for resources).

NOTE:

There are a couple of pitfalls that I will warn you about in this document, so please read THESE instructions, as well as the guides that will be posted to get you through this week's work.

As you go through the guides, remember that versioning is important. Be sure to install the following versions of ansible and kolla-ansible in your virtual environments (do this on management only).

Ansible==2.9

Kolla-ansible==10.2.0

Section 1: Basic Maintenance – Passwordless ssh and virtual environments (10% of grade)

If you did not do so last week, enable passwordless ssh (as root, so you ALSO need to enable/permit root login from ssh) from the management node to every other server.

Recall we covered this in class (**it's in the powerpoint**). The easy way is to use the ssh-copy-id command to do this for you.

You should perform all actions as root and not the regular user. So before starting the instructions in the guide, go ahead and change to the root user.

First of all, be sure to follow the instructions for whichever OS you are using for your servers.

You should have a taste of ansible and python virtual environments from last week. Although we do have a working ceph install, there are some additional things we will need to do to integrate our ceph deployment with OpenStack. We will be using the bootstrap-servers command to prep our servers for the OpenStack install.

To start with, go ahead and create virtual environments for your openstack deployment that contain python3 and updated pip on EVERY server, and make sure to put them in the **SAME PLACE** and name them the **SAME THING** everywhere.

For example, if you make a virtual environment **my_virtual_environment**, and you put it in your root directory **"/root/my_virtual_environment**" on your management node, then your storage, controller, and compute nodes should also have virtual environments at **"/root/my_virtual_environment**".

You should **ONLY** install **ansible** and **kolla-ansible** in your **virtual environment** on your **management** node. Be sure to install the appropriate versions (listed above). You may also want to install the **wheel** package first, as it will want to use that when it's setting up kolla-ansible.

Section 2: Bootstrap File (10%)

Please see the relevant links below for using the bootstrap script. We will be using virtual environments, so you will also need to pay attention to those links, as well.

If you navigate to:

[your virtual environment folder]/share/kolla-ansible/ansible/roles/baremetal/defaults/main.yml

You can configure many bootstrapping options in here so that you don't have to type them in at the CLI when you run the command.

Be sure to AT LEAST set the following:

enable_host_ntp: False ← This will tell the bootstrap playbook to setup ntp for you, but the ceph-ansible playbooks already did this for us.

enable_docker_repo: False ← This will tell the bootstrap playbook NOT to install docker for you. This is not needed since we already did this.

Go ahead and take a look at the other options. There may be others you want to set in here. Be sure to read through the bootstrapping and virtual environment links so that you know what these options mean and how you might want to use them.

For the variable **docker_custom_config**, enter what you wrote for your **daemon.json** file. Even though you've already done this, each time the bootstrap script runs it will modify that file. Leaving it blank, will cause it to erase the contents of that file.

In addition to this, you will need to add one extra property:

"iptables": false

Not doing this will likely cause you to be unable to ssh into your virtual machine from your other servers, as docker adds rules to your iptables that will sometimes conflict with the rules you already have in place (like NAT for instance).

Don't forget to specify where your virtual environment is, so bootstrap knows where to install all of the python packages.

Note:

Make sure to use same name for virtual environment everywhere (see the note in Section 1), and put it in the same location on every server. So, whatever you called your virtual environment on your management node, and wherever you put it, use that same information for your other servers.

Section 3: Docker Images (20% of grade)

Remember, the registry (your repo) is ONLY on the management node.

I have given you a list of docker images that you need to pull.

You should use the tag **ussuri**

I suggest you write a script to do this.

The process should be as follows:

- Pull and image from docker hub
  - docker pull [image_name]:[tag]
- Tag that image
  - docker tag [image_name]:[tag] [registry]:[port #]/[image_name]:[tag]
- Push that image to your repo (your registry)
  - docker push [registry]:[port #]/[image_name]:[tag]
- Once you've pulled all of the images and tagged and pushed them into your registry, verify that your docker environment is configured correctly by manually pulling just **ONE** image from YOUR registry from **EACH** of your servers. The command should look something like below:
  - docker pull [registry]:4000/[some image name]:ussuri
  - You should do this once on EACH of your OTHER servers (NOT mgmt), to make sure they are able to pull images from your registry

Section 4: Integrating Ceph

This week, we need to integrate our existing ceph deployment with OpenStack so that OpenStack is allowed to use ceph. To do that, we need to:

1) Create some pools for use by various openstack services
2) Create authentication keys for “users” (the openstack services) against those pools
3) Pass all of that information on to our kolla-ansible playbooks so that it can do the heavy lifting for us.

For starters, since it looks like the size and min_size may not actually have been set globally for many of us, you can configure global settings for ceph from your controller node. Running the two commands below will set the default size and min_size for any newly created pool to the correct values.

ceph config set global osd_pool_default_min_size 1

ceph config set global osd_pool_default_size 1

Create the following config directories for OpenStack on your **management node**

/etc/kolla/config/cinder

/etc/kolla/config/cinder/cinder-backup

/etc/kolla/config/cinder/cinder-volume

/etc/kolla/config/glance

/etc/kolla/config/gnocchi

/etc/kolla/config/nova

On your **controller** node, you need to create the appropriate pools for openstack to use.

The link below shows you which pools you need to create. It does not show you that you need to create the gnocchi pool. Check out the external ceph section of your globals.yml file to see what it should be called and who the user should be.

Creating Pools in Ceph for Openstack

Don’t forget to initialize all of the pools using rbd.

Once you’ve done that, on your **controller** node, you need to create users in ceph, that are allowed to use those pools. Users are called clients. For each client, you need to create a keyring file for them.

This is also covered in the link above.

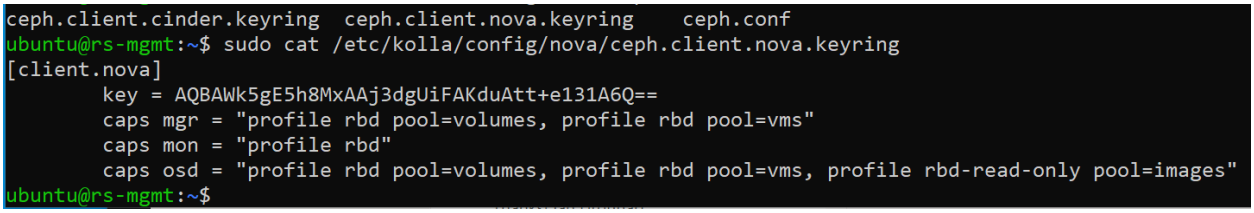
The gnocchi client ONLY needs permission to interact with the gnocchi pool.

Once you've created the appropriate keyrings, you need to export them as a file that you can copy into the kola-ansible config directories you made above (/etc/kola/config/xxx).

You can create the files using the command below:

ceph auth get client.**[user]** -o ceph.client.**[user]**.keyring

The screenshot below is an example of what the contents of one of your folders should be, as well as what the contents of one of the keyring files should look like. Yours should look identical to mine, minus the **key** field.



You can see which pools you need to create (and what the user names should be) in /etc/kolla/globals.yml in the External Ceph section. Be sure that everything matches.

The pools you create should match the pool names in /etc/kolla/globals.yml

The users you create should also match the user.

The keyring files you create should also match the information there.

If you do happen to create pools, or keyring files with different names, just be sure to change what is written in the external ceph section of your globals.yml file so that kolla-ansible can configure everything appropriately. The users, however, do need to match, as kolla-ansible WILL create those users whether you specify them or not. And those users NEED permission to use the ceph pools you created.

You will need to copy some keys into multiple folders. Be sure to check out kolla-ansible's documentation to ensure that you put all of the keyring files (and ceph.conf files) in the appropriate folders (that you made at the beginning of this section).

Kolla-Ansible External Ceph

Section 5: Configuration, Bootstrapping, Pretest, and Pull (50%)

In YOUR /etc/kolla/globals.yml file you should set the following options:

Kolla options

kolla_base_distro: “whatever your operating system is”

kolla_install_type: “binary”

openstack_release: “ussuri”

uncomment ***node_custom_config***

kolla_internal_vip_address: “unused ip address on your **internet** network”

This ip address is the equivalent to the ip address that you all have been forwarding to to log into the horizon dashboard from a web browser (192.168.0.254).

kolla_external_vip_address: “unused ip address on your encrypted network”

Same as above, but this will be used for encrypted traffic.

Docker Options

docker_registry: “ [your registry ip]:4000”

docker_namespace: “kolla”

docker_registry_insecure: ← just uncomment this line

Neutron – Networking Options

network_interface: “br0” ← If you made bridges on all your nodes, including your storage node, this should be the name of your bridge. If you did not do this, you will have to specify each of these devices in your multinode file (inventory file). Both the kolla-ansible quick start and the multimode guides show examples of how to do this.

kolla_external_vip_interface: “ethA” ← this should be the eth device on your second internal network (the “secure” network not NAT) This should be an **INTERFACE NAME** NOT an ip address.

api_interface: “br0” ← see comment above

storage_interface: “ethB” ← This should be the eth device on your storage network. This network is used for ceph traffic. This should be an **INTERFACE NAME** NOT an ip address.

cluster_interface: “ethB” ← storage network. This network is used for ceph replication data. This should be an **INTERFACE NAME** NOT an ip address.

tunnel_interface: “br0” ← see above. Needs internet access.

dns_interface: “br0” ← see above.

neutron_external_interface: “veth1” ← this should be the end of your veth pair that is NOT attached to the bridge. I attached veth0 to my bridge, so I will add veth1 here.

TLS options

The team working on the kolla-ansible scripts have been working very hard! You are now able to enable TLS for internal AND external networks (and backend networks) using kolla-ansible. However, since we will be using the auto-generated self-signed certs, there are still some issues surrounding authenticating with the cli when using self-signed certs, so we will refrain from taking advantage of all of these just yet.

Be sure to set the variable to enable tls on public network ONLY.

Openstack Options

The following openstack options should be uncommented AND set to yes.

enable_:

- openstack_core
- ceilometer
- central_logging
- cinder
- cinder_backup
- openvswitch ← just uncomment this one
- gnocchi
- gnocchi_statsd

disable (set to no):

enable_chrony: “no”

External Ceph Options

external_ceph_cephx_enabled: “yes”

Uncomment other variables except for the two in the manilla section (we won’t be enabling this project in openstack).

Glance image options

glance_backend_ceph: “yes”

glance_backend_file: “no”

Cinder Block Storage Options

cinder_backend_ceph: “yes”

cinder_backup_driver: “ceph”

Gnocchi options

Gnocchi_backend_storage: “ceph”

Nova compute options

nova_backend_ceph: “yes”

nova_console: “novnc”

nova_compute_virt_type: “qemu”

Modifying Playbooks

We need to modify some playbooks so that the OpenStack install will work, as some of the settings don’t quite work as-is.

In the file:

“Roles” (or playbooks for configuring specific OpenStack services) can be found at:

[your virtual environment]share/kolla-ansible/ansible/roles…

You will need to modify the following files:

Neutron Role
File location:

[your virtual environment]/share/kolla-ansible/ansible/roles/neutron/templates/ml2_conf.ini.j2

At the top of this file, under the section titled [ml2]:
path_mtu = 1400 ← need to tell openstack what your system’s mtu is, so that neutron can appropriately adjust for packet sizes

Neutron Role
File path:

[your virtual environment]/share/kolla-ansible-ansible/roles/neutron/templates/neutron.conf.j2

In the DEFAULT section add the following:
global_physnet_mtu = 1400 ← same purpose as previous similar statement. Need to tell openstack what your system’s default mtu is.

Operating System Version

File path:

[virtual env folder]/share/kolla-ansible/ansible/roles/prechecks/vars/main.yml

While this version of OpenStack "will" run in ubuntu 20. These playbooks were created and tested on Ubuntu 18. For that reason, they don't have ubuntu 20 in the list of supported operating systems for their playbooks. However, the playbooks **WILL** create a working OpenStack Cloud.

You will need to add focal to the list of host_os_distributions variable, like below:

host_os_distributions

- "focal"

Running playbooks

At this point, you are nearly ready to bootstrap your servers, do your prechecks, and pull docker images. You first need to set up your inventory file so that it contains information about all of your machines. The multinode file that you copied into your home directory earlier is where you do this.

The following groups should only contain the hostname for your controller node:

control, network, monitoring

The compute group should have the hostname of your compute node.

The storage group should have the hostname of your storage node.

If you do not have uniform interface device names across all of your servers, in addition to specifying the hostname of a machine, you will also have to specify an interface to use for EACH of the networking interfaces listed in the globals.yml file that is NOT the SAME for EVERY server. Below is an example of how you could do this for a single entry:

[network]
controller network_interface=br0 kolla_external_vip_interface=eth1 api_interface=br0 storage_interface=eth2 cluster_interface=eth2 tunnel_interface=br0 dns_interface=br0

You should now be ready to Bootstrap, Precheck, and Pull images for your OpenStack Deployment.

Note: You need to tell kolla-ansible that you're using a virtual environment. This is covered in the links below, but I'll mention it here anyway.

When you are issuing kolla-ansible commands,

(kolla-ansible -i multinode prechecks) ← for example

You should add an additional command-line option that tells kolla-ansible where the python executable is located in your virtual environment. This is the same as the

way we had to pass in what type of docker we were using for ceph-ansible last week. For example, with the command above, you might add:

kolla-ansible -i multinode **-e ansible_python_interpreter=[virtual environment path]/bin/python** prechecks

You should be ready to run the bootstrap-servers command now.

DO NOT USE THE -e flag for bootstrapping.

BOOTSTRAP NOW

Once you've bootstrapped your servers, you can do a precheck, just to check for any easy-to-catch errors. Before you do this, be sure to generate passwords for all of the services. You can choose to manually populate /etc/kolla/passwords.yml file with passwords for all, but its easier to just use the kolla-genpwd command.

You should be ready to run the prechecks command now.

USE THE -e flag for prechecks so kolla-ansible knows to use the python from your virtual environment

PRECHECKS NOW

Note: You might want to consider installing the expect package so that you can use the unbuffer utility.

During the course of the precheck (and your actual install), you may have around 1000+ lines of text come across your screen (and it may take 30 min – 1 hour to complete). You can use the following command (if you install the expect package) to write all of that output to a file AND show it on the console at the same time. This way, if there is an error that is too far back for you to scroll to (and you weren't standing in front of your screen the entire time), you can at least look inside your file to find the error. Don't forget to add the path to your python interpreter in your virtual environment.

unbuffer kolla-ansible -i multinode prechecks 2>&1 | tee [some filename to save it too]

You may also want to consider using some kind of virtual terminal application (like **screen**) when you're doing this section. Screen should already be installed in the operating system. Screen is kind of the terminal equivalent of how windows or mac can have multiple "virtual" desktops. The nice thing about screen is that if you get disconnected for some reason, it will continue to execute whatever tasks are running. When you reconnect, you can simply re-attach to the screen and continue wherever it left off.

Once you've passed all of your prechecks you should now be ready to execute the pull command.

USE THE -e flag for pull so kolla-ansible knows to use the python from your virtual environment

PULL NOW (KOLLA-ANSIBLE PULL, YOU SHOULD HAVE ALREADY DONE THE DOCKER PULL)

Once you have successfully pulled, **STOP. DO NOT DEPLOY YET!** Let the TA know when you finish this so that he can grade it before you move on. If you submit this before the Sunday night due date, go ahead and submit everything to canvas and email the TA so he can check and grade it and you can potentially get a head start on next week's portion. This would also be a good time to take snapshots of all of your vm volumes.

Deliverables

Section I: Basic Maintenance 10%

- Passwordless ssh enabled for root account on management to every other server
- Basic dependencies installed on all servers.
- Virtual environments created on all servers. Named the same. In the same location.

Section II: Bootstrap 10%

- Bootstrap config file set NOT to configure ntp
- Bootstrap config file variable set to path to your virtual environment
- Bootstrap config file set to correctly configure mtu, insecure registry, and iptables info in /etc/docker/daemon.json for OTHER servers (you already did this manually for the last milestone, but it will erase that if you don't put something).

Section III: Docker 15%

- All docker images pulled on **management** and exist in private registry on **management** node
- Other servers are able to pull images from private registry on management node

Section IV: Ceph 15%

- Appropriate pools created and initialized via rbd in ceph for use in openstack
- Appropriate users created in ceph. Users have correct permissions to interact with their pools

- Appropriate keyring files generated and copied into the correct directory in /etc/kolla/config/

Section V: Config 50%

- All config files modified appropriately
- All playbook modifications are correct
- Successful bootstrap
- Successful prechecks
- Successful pull

Guides

1) Kolla-Ansible Quick Start Guide
2) Kolla-Ansible Multinode Guide
3) Kolla-Ansible TLS for External Network
4) Kolla-Ansible Bootstrap Servers
5) Kolla-Ansible Virtual Environments
6) Kolla-Ansible External Ceph
7) Creating Pools in Ceph for Openstack
8) Creating Keyrings for Users in Ceph (to be used with certain pools
9) Managing Users/Credentials in Ceph (making/remaking/deleting users – keyrings)