INSTALLATION AND CONFIGURATION OF OPENSTACK AND SAHARA TO CREATE HADOOP CLUSTERS

Deploying a Hadoop cluster for test purposes using OpenStack with Sahara involves several methodical steps. As a widely accepted open-source platform, OpenStack integrates Sahara to streamline the provision and management of Hadoop clusters. This process for creating a core Hadoop cluster includes:

Preparation of the environment:

It is necessary to ensure that the OpenStack environment is fully operational, including services such as Keystone, Glance, Nova, Neutron, and optionally Cinder and Swift.

Verifying the availability of adequate resources in the OpenStack cloud is essential to support the Hadoop cluster.

Installation and configuration of the Sahara service in the OpenStack cluster is required. The Sahara simplifies provisioning and scaling of Hadoop clusters.

Sahara Image registration:

Acquiring a pre-built image compatible with the Sahara, Hadoop or creating one according to Sahara Image Elements documentation is crucial.

The Hadoop image must be uploaded to Glance, either via the OpenStack dashboard or the command line.

Registering the image in the Sahara with appropriate Hadoop version tags and other relevant metadata is necessary.

Network configuration:

The creation of a private network and subnetwork in Neutron exclusively for the Hadoop cluster is important for node communication.

Configuring network settings to allow basic port communication involves setting up groups and security policies.

Create a Saharan cluster template:

Creating a cluster template in the Sahara is vital. This template outlines the configuration of the Hadoop cluster, including the Hadoop version, the node groups, and the scents of the specimens.

The determination of the number of specimens for each group of assemblies is based on testing requirements.

Data source and project binaries:

Uploading the necessary data sources to test data processing is important either in the internal Sahara database or in an external storage system.

Uploading binaries or job scripts is required to test specific Hadoop jobs.

Starting the Hadoop cluster:

The cluster is initiated using the predefined template and network configurations.

Monitoring the assurance process through the Sahara Dashboard or CLI is essential to ensure proper setup.

Testing and validation:

Checking that Hadoop services work on all nodes can be done through Hadoop web interfaces or command-line tools.

The implementation of the main tasks of Hadoop is necessary to confirm the functionality of the cluster, either directly or through the Sahara.

Monitoring and logging:

Implementing monitoring for the Hadoop cluster using OpenStack's Ceilometer and Aodh is critical to track resource use and health.

Setting up logging to collect and analyze Hadoop logs helps troubleshoot and optimize. These logs can be stored in Swift for persistence and accessibility.

Preparation of the system

The first step in preparing the system involves updating the package index to ensure that all software is up to date. This can be done by implementing:

sudo apt update

It is then essential to install a database server. MySQL or MariaDB servers are a good choice for this purpose. The installation process in the system is initiated with the following commands:

sudo apt install -y mariadb-server

sudo systemctl start mariadb

sudo systemctl enable mariadb

After installing the database server, installation protection is a critical step. This includes setting a password and implementing other security measures. The following command is used to protect the MariaDB installation:

sudo mysql_secure_installation

The next phase involves the creation of the necessary database for Keystone, the OpenStack identity service. This includes creating a special database, configuring a user, and granting appropriate privileges. These steps ensure that Keystone has the necessary database access and permissions to function properly. The following commands are executed in the MySQL shell to achieve this:

sudo mysql -u root -p

CREATE DATABASE keystone;

GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' IDENTIFIED BY 'KEYSTONE_DBPASS';

GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' IDENTIFIED BY 'KEYSTONE_DBPASS';

FLUSH PRIVILEGES;

EXIT;

Installing the Keystone service

The process begins with the installation of Keystone, which is the component of the OpenStack identity service. This can be achieved using the following command:

sudo apt install -y keystone

Once Keystone is installed, configuring the Keystone configuration file is the next important step. This includes editing /etc/keystone/keystone.conf to set up the database connection. Specifically, the connection string must be specified in the configuration file:

connection = mysql+pymysql://keystone:KEYSTONE_DBPASS@controller/keystone

In addition, in the [token] section of the same configuration file, it is essential to specify the token provider as follows:

provider = fernet

The next step involves synchronizing the database to ensure that the Keystone database schema is up to date:

sudo keystone-manage db_sync

Fernet keys are used to generate tokens in Keystone. Initializing these keys is a critical security step:

sudo keystone-manage fernet_setup --keystone-user keystone --keystone-group keystone

sudo keystone-manage credential_setup --keystone-user keystone --keystone-group keystone

To initialize the Keystone service with default values and to create an administrative user, the following bootstrap command is used:

```
sudo keystone-manage bootstrap --bootstrap-password ADMIN_PASS \
        --bootstrap-admin-url http://controller:35357/v3/ \
        --bootstrap-internal-url http://controller:35357/v3/ \
        --bootstrap-public-url http://controller:5000/v3/ \
        --bootstrap-region-id RegionOne
```

Here, we assume that the "controller" points to the local machine (or where Keystone is installed).

After configuring Keystone, it is necessary to restart the Apache server to apply the changes:

```
sudo systemctl restart apache2
```

Then we need to set up environmental variables that will be used to interact with the service. These environment variables typically include credentials and endpoint information that allow the openstack command-line client and other clients to communicate with Keystone for authentication and service discovery.

In the admin-openrc file we can set some environment variables:

```
export OS_PROJECT_DOMAIN_NAME=Default
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=ADMIN_PASS
export OS_AUTH_URL=http://controller:5000/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
```

To load these variables, we must execute:

source admin-openrc

Finally, to check that everything is set up correctly and the Keystone functions as expected, the following command can be used to issue tokens:

openstack token issue

Installing the Glance service

First, we install the Glance packages using apt:

sudo apt install -y glance

The first step could be to set up the database in MariaDB:

sudo mysql -u root -p

CREATE DATABASE glance;

GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' IDENTIFIED BY 'GLANCE_DBPASS';

GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%' IDENTIFIED BY 'GLANCE_DBPASS';

FLUSH PRIVILEGES;

EXIT;

We need to edit multiple configuration files to configure Glance to use the Identity Service and determine where it will store the images.

Edit at /etc/glance/glance-api.conf. We set the database connection string, the Keystone identity service and server are configured:

[database]

connection = mysql+pymysql://glance:GLANCE_DBPASS@controller/glance

[keystone_authtoken]

www_authenticate_uri = http://controller:5000

auth_url = http://controller:5000

memcached_servers = controller:11211

auth_type = password

project_domain_name = default

user_domain_name = default

project_name = service

username = glance

password = GLANCE_PASS

[glance_store]

stores = file,http

default_store = file

filesystem_store_datadir = /var/lib/glance/images/

Edit at /etc/glance/glance-registry.conf. We set the database connection and configure the keystone identity service.

[database]

connection = mysql+pymysql://glance:GLANCE_DBPASS@controller/glance

[keystone_authtoken]

www_authenticate_uri = http://controller:5000

auth_url = http://controller:5000

memcached_servers = controller:11211

auth_type = password

project_domain_name = default

user_domain_name = default

project_name = service

username = glance

password = GLANCE_PASS

Sync the database for Glance:

sudo glance-manage db_sync

Create a Glance user in Keystone:

openstack user create --domain default --password-prompt glance

Add the administrator role to the Glance user:

openstack role add --project service --user glance admin

Create the Glance service object:

openstack service create --name glance --description "OpenStack Image" image

Creating Service API Points:

openstack endpoint create --region RegionOne image public http://controller:9292

openstack endpoint create --region RegionOne image internal http://controller:9292

openstack endpoint create --region RegionOne image admin http://controller:9292

Install Nova service

First, the creation of a database and user for Nova in MariaDB is needed:

sudo mysql -u root -p

CREATE DATABASE nova_api;

CREATE DATABASE nova;

CREATE DATABASE nova_cell0;

GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'localhost' IDENTIFIED BY 'NOVA_DBPASS';

GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'%' IDENTIFIED BY 'NOVA_DBPASS';

GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' IDENTIFIED BY 'NOVA_DBPASS';

GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' IDENTIFIED BY 'NOVA_DBPASS';

GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova'@'localhost' IDENTIFIED BY 'NOVA_DBPASS';

GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova'@'%' IDENTIFIED BY 'NOVA_DBPASS';

FLUSH PRIVILEGES;

EXIT;

Installing Nova packages:

sudo apt install nova-api nova-conductor nova-novncproxy nova-scheduler

The file /etc/nova/nova.conf is edited with the following configurations:

[api_database]

connection = mysql+pymysql://nova:NOVA_DBPASS@controller/nova_api

```
[database]
connection = mysql+pymysql://nova:NOVA_DBPASS@controller/nova

[DEFAULT]
transport_url = rabbit://openstack:RABBIT_PASS@controller
auth_strategy = keystone
my_ip = CONTROLLER_IP
enabled_apis = osapi_compute,metadata

[keystone_authtoken]
auth_url = http://controller:5000/v3
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = nova
password = NOVA_PASS

[vnc]
enabled = true
server_listen = $0.0.0.0
server_proxyclient_address = $CONTROLLER_IP

[glance]
```

api_servers = http://controller:9292

[oslo_concurrency]

lock_path = /var/lib/nova/tmp

[placement]

region_name = RegionOne

project_domain_name = Default

project_name = service

auth_type = password

user_domain_name = Default

auth_url = http://controller:5000/v3

username = placement

password = PLACEMENT_PASS

[scheduler]

discover_hosts_in_cells_interval = 300

Fill in the Nova databases:

su -s /bin/sh -c "nova-manage api_db sync" nova

su -s /bin/sh -c "nova-manage cell_v2 map_cell0" nova

su -s /bin/sh -c "nova-manage cell_v2 create_cell --name=cell1 --verbose" nova

su -s /bin/sh -c "nova-manage db sync" nova

It should be checked that the cells are registered correctly:

nova-manage cell_v2 list_cells

The services of Nova are restarted:

sudo systemctl restart nova-api.service nova-consoleauth.service nova-scheduler.service nova-conductor.service nova-novncproxy.service

List of components of the Nova successful installation verification service:

openstack compute service list

The Placement service has been taken out of Nova and has now been a standalone service for some time.

sudo apt install placement-api

The Placement service requires its own database setup in MariaDB:

sudo mysql -u root -p

CREATE DATABASE placement;

GRANT ALL PRIVILEGES ON placement.* TO 'placement'@'localhost' IDENTIFIED BY 'PLACEMENT_DBPASS';

GRANT ALL PRIVILEGES ON placement.* TO 'placement'@'%' IDENTIFIED BY 'PLACEMENT_DBPASS';

FLUSH PRIVILEGES;

EXIT;

Edit the Placement configuration file /etc/placement/placement.conf:

[placement_database]

connection = mysql+pymysql://placement:PLACEMENT_DBPASS@controller/placement

[api]

auth_strategy = keystone

[keystone_authtoken]

auth_url = http://controller:5000/v3

memcached_servers = controller:11211

auth_type = password

project_domain_name = Default

user_domain_name = Default

project_name = service

username = placement

password = PLACEMENT_PASS

Filling in the Placement database:

su -s /bin/sh -c "placement-manage db sync" placement

Restart the Placement service:

sudo systemctl restart apache2

Finally, we can use the OpenStack CLI to check if Nova and Placement services are working:

openstack compute service list

openstack placement service list

The transport_url setting is used to configure the message queue server for OpenStack services, which is RabbitMQ by default. This setup is crucial for the operation of your OpenStack environment as it facilitates communication between different components. If RabbitMQ is not yet installed and configured on our controller node:

sudo apt update

sudo apt install rabbitmq-server

sudo systemctl enable rabbitmq-server

sudo systemctl start rabbitmq-server

Once RabbitMQ is running, you will need to create a user for OpenStack with the necessary permissions.

sudo rabbitmqctl add_user openstack RABBIT_PASS

sudo rabbitmqctl set_permissions openstack ".*" ".*" ".*"

The Nova service should be registered:

openstack service create --name nova --description "OpenStack Compute" compute

Creating public, internal and admin endpoints for Nova:

openstack endpoint create --region RegionOne compute public http://PUBLIC_ENDPOINT

openstack endpoint create --region RegionOne compute internal http://INTERNAL_ENDPOINT

openstack endpoint create --region RegionOne compute admin http://ADMIN_ENDPOINT

The following is a check of the creation of the services:

openstack service list

openstack endpoint list

Create a Nova user

openstack user create --domain default --password NOVA_PASS nova

Add the admin role to the Nova user

openstack role add --project service --user nova admin

Create a Nova Service Object

openstack service create --name nova --description "OpenStack Compute" compute

Installing Neutron Service

The first step is the creation of a Neutron database and suitable access for a Neutron user:

mysql -u root -p

CREATE DATABASE neutron;

GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost' IDENTIFIED BY 'NEUTRON_DBPASS';

GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%' IDENTIFIED BY 'NEUTRON_DBPASS';

FLUSH PRIVILEGES;

EXIT;

Establishment of Neutron Service Credentials:

openstack user create --domain default --password-prompt neutron

openstack role add --project service --user neutron admin

openstack service create --name neutron --description "OpenStack Networking" network

Creation of Neutron Service API points:

openstack endpoint create --region RegionOne network public http://controller:9696

openstack endpoint create --region RegionOne network internal http://controller:9696

openstack endpoint create --region RegionOne network admin http://controller:9696

Installing the Neutron packages:

sudo apt install neutron-server neutron-plugin-ml2 neutron-linuxbridge-agent neutron-l3-agent neutron-dhcp-agent neutron-metadata-agent

We edit the file /etc/neutron/neutron.conf:

[database]

connection = mysql+pymysql://neutron:NEUTRON_DBPASS@controller/neutron

Configure access to the message queue RabbitMQ:

[DEFAULT]

transport_url = rabbit://openstack:RABBIT_PASS@controller

Configuring the Keystone Authentication Service:

[keystone_authtoken]

auth_url = http://controller:5000

memcached_servers = controller:11211

auth_type = password

project_domain_name = default

user_domain_name = default

project_name = service

username = neutron

password = NEUTRON_PASS

Configure the Modular Layer 2 plugin (ML2) by editing /etc/neutron/plugins/ml2/ml2_conf.ini:

[ml2]

type_drivers = flat,vlan,vxlan

tenant_network_types = vxlan

mechanism_drivers = linuxbridge,l2population

[ml2_type_flat]

flat_networks = provider

[ml2_type_vlan]

network_vlan_ranges = provider:100:200

[ml2_type_vxlan]

vni_ranges = 1:1000

[securitygroup]

enable_ipset = true

Completion of the Neutron database:

su -s /bin/sh -c "neutron-db-manage --config-file /etc/neutron/neutron.conf --config-file /etc/neutron/plugins/ml2/ml2_conf.ini upgrade head" neutron

Restart Neutron Service:

sudo systemctl restart neutron-server neutron-linuxbridge-agent neutron-dhcp-agent neutron-metadata-agent

Once the installation is complete, it is appropriate to check that the Neutron service is working:

openstack network agent list

Installing Cinder service

Creation of the Cinder database:

CREATE DATABASE cinder;

GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'localhost' IDENTIFIED BY 'CINDER_DBPASS';

GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'%' IDENTIFIED BY 'CINDER_DBPASS';

Create the Cinder user and add roles:

openstack user create --domain default --password-prompt cinder

openstack role add --project service --user cinder admin

Creating Cinder Service Objects:

openstack service create --name cinderv3 --description "OpenStack Block Storage" volumev3

We create Cinder's access points:

openstack endpoint create --region RegionOne volumev3 public http://controller:8776/v3/%\(tenant_id\)s

openstack endpoint create --region RegionOne volumev3 internal http://controller:8776/v3/%\(tenant_id\)s

openstack endpoint create --region RegionOne volumev3 admin http://controller:8776/v3/%\(tenant_id\)s

Installing Cinder packages:

sudo apt install cinder-api cinder-scheduler python3-cinderclient

We edit /etc/cinder/cinder.conf to configure database access, RabbitMQ message queue access, and other necessary settings:

[database]

connection = mysql+pymysql://cinder:CINDER_DBPASS@controller/cinder

[DEFAULT]

transport_url = rabbit://openstack:RABBIT_PASS@controller

auth_strategy = keystone

my_ip = CONTROLLER_IP

enabled_backends = lvm

[keystone_authtoken]

auth_uri = http://controller:5000

auth_url = http://controller:35357

memcached_servers = controller:11211

auth_type = password

project_domain_name = default

user_domain_name = default

project_name = service

username = cinder

password = CINDER_PASS

[lvm]

volume_driver = cinder.volume.drivers.lvm.LVMVolumeDriver

volume_group = cinder-volumes

iscsi_protocol = iscsi

iscsi_helper = tgtadm

We complete the Cinder database:

su -s /bin/sh -c "cinder-manage db sync" cinder

Cinder's services should be restarted:

sudo service cinder-scheduler restart

sudo service cinder-api restart

We check the correctness of the installation:

openstack volume service list

The next thing to do is to set up the LVM. First, we check if we have lvm2 installed:

sudo apt-get install lvm2

Physical volume will be required for LVM. This can be a partition, an entire disk or a RAID array. Here we conditionally assume that it is called sdX.

sudo pvcreate /dev/sdX

Cinder will use an LVM volume group to provide a logical volume for block storage. Creating a group is done with the following command:

sudo vgcreate cinder-volumes /dev/sdX

We need to verify that the iSCSI Target User Space Tool (tgtadm) is installed, which is used by the LVM driver to export block devices through iSCSI.

sudo apt-get install tgt

After the configuration is changed, the cinder-volume service must be restarted to apply the changes:

sudo systemctl restart openstack-cinder-volume

Installing Horizon

Installing Horizon, OpenStack's dashboard, involves several steps to set up the environment, configure the necessary components, and ensure it integrates properly with existing OpenStack services.

Installing the "Horizon" package:

sudo apt install openstack-dashboard

First, you need to configure Horizon settings in /etc/openstack-dashboard/local_settings.py:

OPENSTACK_HOST = "controller"

```
CACHES = {
    'default': {
        'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
        'LOCATION': 'controller:11211',
    },
}
TIME_ZONE = "Europe/Sofia"
```

Reload the web server:

sudo systemctl reload apache2

Installing Sahara

Creation of the database and the database user in Sahara:

mysql -u root -p

CREATE DATABASE sahara;

GRANT ALL PRIVILEGES ON sahara.* TO 'sahara'@'localhost' IDENTIFIED BY 'SAHARA_DBPASS';

GRANT ALL PRIVILEGES ON sahara.* TO 'sahara'@'%' IDENTIFIED BY 'SAHARA_DBPASS';

FLUSH PRIVILEGES;

EXIT;

Installing the packages from Sahara:

sudo apt-get install sahara sahara-api sahara-engine python3-saharaclient

It is necessary to edit the file /etc/sahara/sahara.conf with the following directives:

[database]

connection = mysql+pymysql://sahara:SAHARA_DBPASS@controller/sahara

[DEFAULT]

debug = false

auth_strategy = keystone

osapi_sahara_listen = 0.0.0.0

osapi_sahara_listen_port = 8386

[keystone_authtoken]

auth_url = http://controller:5000/v3

username = sahara

password = SAHARA_PASS

user_domain_name = Default

project_name = service

project_domain_name = Default

Database synchronization:

sahara-db-manage --config-file /etc/sahara/sahara.conf upgrade head

Registering Sahara in Keystone:

openstack user create --domain default --password-prompt sahara

openstack role add --project service --user sahara admin

openstack service create --name sahara --description "Sahara Data Processing" data-processing

Creating the service API endpoints in Sahara:

openstack endpoint create --region RegionOne data-processing public http://controller:8386/v1.1/%\(tenant_id\)s

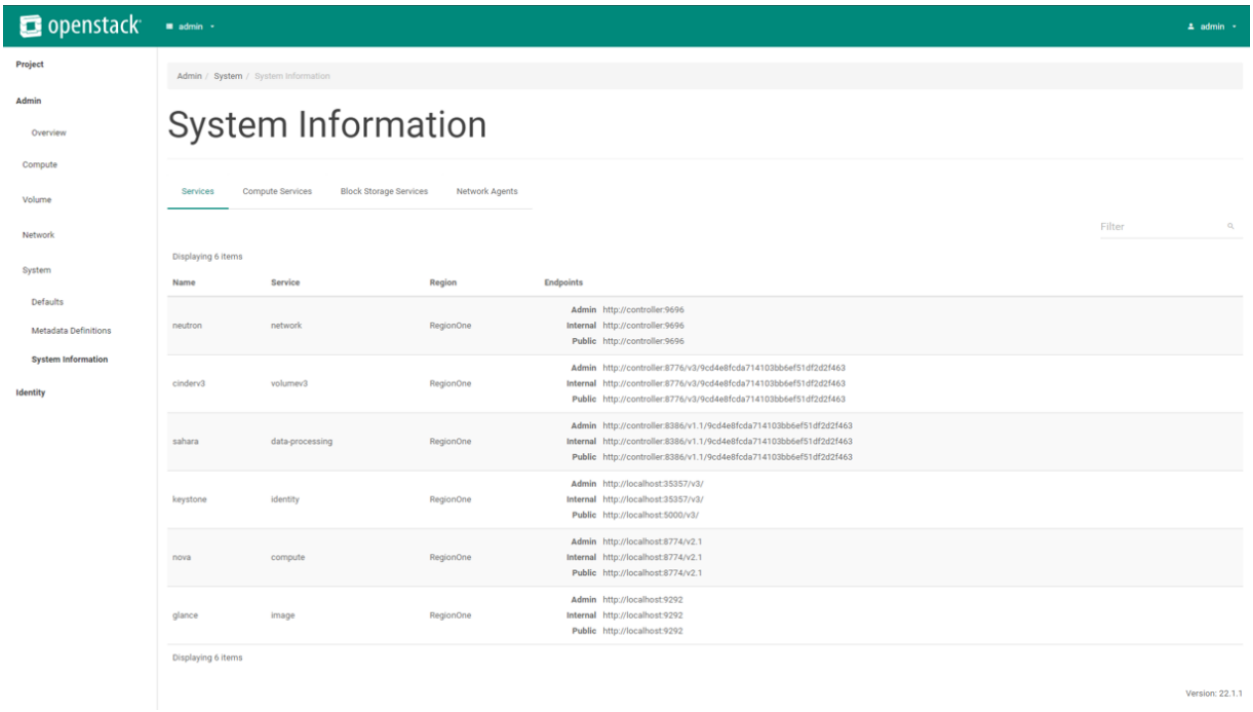
openstack endpoint create --region RegionOne data-processing internal http://controller:8386/v1.1/%\(tenant_id\)s

openstack endpoint create --region RegionOne data-processing admin http://controller:8386/v1.1/%\(tenant_id\)s

Finally, you need to restart the services in Sahara:

service sahara-engine restart

With successful installation and configuration of all services, in Horizon you can see the working services in the current installation of OpenStack:



References:

1. OpenStack; OpenStack Installation Guide; https://docs.openstack.org/install-guide/
2. OpenStack; Sahara Installation Guide; https://docs.openstack.org/sahara/latest/install/installation-guide.html
3. Ubuntu; Learn about OpenStack services and their functions; https://ubuntu.com/tutorials/learn-about-openstack-services-and-their-functions#1-overview
4. Red Hat; Understanding OpenStack; https://www.redhat.com/en/topics/openstack
5. OpenStack; Sahara (Data Processing) UI User Guide; https://docs.openstack.org/sahara/ocata/horizon/dashboard.user.guide.html