INSTALLATION AND CONFIGURATION OF OPENSTACK AND SAHARA TO CREATE HADOOP CLUSTERS

ИНСТАЛАЦИЯ И КОНФИГУРАЦИЯ НА OPENSTACK И SAHARA ЗА СЪЗДАВАНЕ НА HADOOP КЛЪСТЕРИ

Разполагането на клъстер Hadoop за тестови цели с помощта на OpenStack със Sahara включва няколко методически стъпки. Като широко приета платформа с отворен код, OpenStack интегрира Sahara, за да рационализира осигуряването и управлението на клъстерите Hadoop. Този процес за създаване на основен клъстер Hadoop включва:

Подготовка на околната среда:

Необходимо е да се гарантира, че средата на OpenStack е напълно функционираща, включително услуги като Keystone, Glance, Nova, Neutron и по желание Cinder и Swift.

Проверката на наличието на адекватни ресурси в облака на OpenStack е от съществено значение за поддръжката на клъстера Hadoop.

Необходима е инсталация и конфигуриране на услугата Sahara в клъстера OpenStack. Сахара опростява провизирането и мащабирането на клъстерите Hadoop.

Сахара Регистрация на изображението:

Придобиването на предварително изградено изображение, съвместимо със Сахара, Hadoop или създаването на такова според документацията на Sahara Image Elements е от решаващо значение.

Изображението на Hadoop трябва да бъде качено в Glance, или чрез таблото за управление на OpenStack, или чрез командния ред.

Регистрирането на изображението в Сахара с подходящи тагове за версията на Hadoop и други подходящи метаданни е необходимо.

Мрежова конфигурация:

Създаването на частна мрежа и подмрежа в Neutron изключително за клъстера Hadoop е важно за комуникацията на възлите.

Конфигурирането на мрежовите настройки, за да се позволи основна комуникация на портовете, включва настройване на групи и правила за защита.

Създаване на шаблон за клъстер в Сахара:

Създаването на клъстерен шаблон в Сахара е от жизненоважно значение. Този шаблон очертава конфигурацията на клъстера Hadoop, включително версията на Hadoop, групите възли и ароматите на екземплярите.

Определянето на броя на екземплярите за всяка група възли се основава на изискванията за тестване.

Източник на данни и двоични файлове за проекти:

Качването на необходимите източници на данни за тестване на обработката на данни е важно или във вътрешната база данни на Сахара, или във външна система за съхранение.

Качването на двоични файлове или скриптове на задания е необходимо за тестване на конкретни задания на Hadoop.

Стартиране на клъстера Hadoop:

Клъстерът се инициира с помощта на предварително дефинирания шаблон и мрежови конфигурации.

Мониторингът на процеса на осигуряване чрез таблото за управление на Сахара или CLI е от съществено значение, за да се гарантира правилната настройка.

Тестване и валидиране:

Проверката, че услугите на Hadoop работят на всички възли, може да се извърши чрез уеб интерфейси на Hadoop или инструменти от командния ред.

Изпълнението на основните задачи на Hadoop е необходимо, за да се потвърди функционалността на клъстера, директно или чрез Сахара.

Мониторинг и регистриране:

Прилагането на мониторинг за клъстера Hadoop с помощта на Ceilometer и Aodh на OpenStack е от решаващо значение за проследяване на използването на ресурсите и здравето.

Настройването на регистрирането за събиране и анализиране на регистрационните файлове на Hadoop помага за отстраняване на неизправности и оптимизация. Тези регистрационни файлове могат да се съхраняват в Swift за постоянство и достъпност.

Подготовка на системата

Първата стъпка в подготовката на системата включва актуализиране на индекса на пакета, за да се гарантира, че целият софтуер е актуален. Това може да стане чрез изпълнение:

sudo apt update

След това е от съществено значение да инсталирате сървър за база данни. MySQL или MariaDB сървърите са подходящ избор за тази цел. Инсталационният процес в системата се инициира със следните команди:

sudo apt install -y mariadb-server

sudo systemctl start mariadb

sudo systemctl enable mariadb

След инсталирането на сървъра на базата данни, защитата на инсталацията е критична стъпка. Това включва задаване на парола и прилагане на други мерки за сигурност. Следната команда се използва за защита на инсталацията на MariaDB:

sudo mysql_secure_installation

Следващата фаза включва създаването на необходимата база данни за Keystone, услугата за идентичност OpenStack. Това включва създаване на специална база данни, конфигуриране на потребител и предоставяне на подходящи привилегии. Тези стъпки гарантират, че Keystone има необходимия достъп до базата данни и разрешения, за да функционира правилно. Следните команди се изпълняват в MySQL обвивката, за да се постигне това:

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;

Инсталиране на Keystone услугата

Процесът започва с инсталирането на Keystone, който е компонентът на услугата за идентичност на OpenStack. Това може да се постигне с помощта на следната команда:

sudo apt install -y keystone

След като Keystone е инсталиран, конфигурирането на конфигурационния файл на Keystone е следващата важна стъпка. Това включва редактиране /etc/keystone/keystone.conf за настройване на връзката с базата данни. По-конкретно, низът за връзка трябва да бъде зададен в конфигурационния файл:

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

Освен това в раздела [token] на същия конфигурационен файл е от съществено значение да се посочи доставчика на токъни, както следва:

provider = fernet

Следващата стъпка включва синхронизиране на базата данни, за да се гарантира, че схемата на базата данни на Keystone е актуална:

sudo keystone-manage db_sync

Fernet ключовете се използват за генериране на токени в Keystone. Инициализирането на тези ключове е критична стъпка за защита:

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

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

За да се инициализира услугата Keystone със стойности по подразбиране и да се създаде административен потребител, се използва следната команда bootstrap:

```
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
```

Тук приемаме, че "контролерът" сочи към локалната машина (или където е инсталиран Keystone).

След конфигурирането на Keystone е необходимо да се рестартира сървъра на Apache, за да приложите промените:

```
sudo systemctl restart apache2
```

След това трябва да настроим променливи на околната среда, които ще се използват за взаимодействие с услугата. Тези променливи на средата обикновено включват идентификационни данни и информация за крайната точка, които позволяват на клиента на командния ред на openstack и други клиенти да комуникират с Keystone за удостоверяване и откриване на услуги.

Във файла admin-openrc можем да зададем някои променливи на средата:

```
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

За да заредим тези променливи, трябва да изпълним:

source admin-openrc

И накрая, за да се провери дали всичко е настроено правилно и Keystone функционира според очакванията, следната команда може да се използва за издаване на токън:

openstack token issue

Инсталиране на Glance услугата

Първо инсталираме пакетите Glance с помощта на apt:

sudo apt install -y glance

Първата стъпка може да бъде настройването на базата данни в 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;

Трябва да редактираме няколко конфигурационни файла, за да конфигурираме Glance да използва услугата за самоличност и да определим къде ще съхранява image-ите.

Следва редактиране на /etc/glance/glance-api.conf. Задаваме низа за връзка с базата данни, конфигурира се услугата за самоличност на Keystone и сървъра:

[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/

Следва редактиране на /etc/glance/glance-registry.conf. Задаваме връзката с базата данни и конфигурираме услугата за самоличност на keystone.

[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

Синхронизиране на базата данни за Glance:

sudo glance-manage db_sync

Създаване на Glance потребител в Keystone:

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

Добавяне на ролята на администратор към потребителя на Glance:

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

Създаване на обекта на услугата Glance:

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

Създаване на точки на API на услугата:

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

Инсталиране на Nova услуга

Първо, необходимо е създаването на база данни и потребител за Nova в MariaDB:

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;

Инсталиране на пакети Nova:

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

Редактира се файла /etc/nova/nova.conf със следните конфигурации:

[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
```

Попълват се базите данни на Nova:

```
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

Следва да се провери дали клетките са регистрирани правилно:

nova-manage cell_v2 list_cells

Рестартират се услугите на Nova:

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

Списък на компонентите на услугата за проверка на успешната инсталация на Nova:

openstack compute service list

Услугата Placement е извадена от Nova и от известно време вече е самостоятелна услуга.

sudo apt install placement-api

Услугата Placement изисква собствена настройка на база данни в 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;

Редактиране на Placement конфигурационния файл /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

Попълване на Placement базата данни:

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

Рестартиране на Placement услугата:

sudo systemctl restart apache2

И накрая, можем да използваме OpenStack CLI, за да проверим дали услугите на Nova и Placement работят:

openstack compute service list

openstack placement service list

Настройката transport_url се използва за конфигуриране на сървъра на опашката за съобщения за услугите на OpenStack, който по подразбиране е RabbitMQ. Тази настройка е от решаващо значение за работата на вашата OpenStack среда, тъй като улеснява комуникацията между различните компоненти. Ако RabbitMQ все още не е инсталиран и конфигуриран на нашия възел на контролера:

sudo apt update

sudo apt install rabbitmq-server

sudo systemctl enable rabbitmq-server

sudo systemctl start rabbitmq-server

След като RabbitMQ работи, ще трябва да се създаде потребител за OpenStack с необходимите разрешения.

sudo rabbitmqctl add_user openstack RABBIT_PASS

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

Следва да се регистрира услугата Nova:

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

Създаване на публични, вътрешни и администраторски крайни точки за 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

Следва проверка на създаването на услугите:

openstack service list

openstack endpoint list

Създаване на потребител на Nova

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

Добавяне на ролята на администратор към потребителя на Nova

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

Създаване на обект на услуга на Nova

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

Инсталиране на Neutron услуга

Първата стъпка е създаването на Neutron база данни и подходящ достъп за Neutron потребител:

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;

Създаване на 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

Създаване на Neutron Service API точки:

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

Инсталиране на Neutron пакетите:

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

Редактираме файла /etc/neutron/neutron.conf:

[database]

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

Конфигуриране на достъпа до опашката за съобщения на RabbitMQ:

[DEFAULT]

transport_url = rabbit://openstack:RABBIT_PASS@controller

Конфигуриране на услугата за удостоверяване на Keystone:

[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

Конфигуриране на приставката Modular Layer 2 (ML2) чрез редактиране /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

Попълване на Neutron базата данни:

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

Рестартиране на Neutron услугата:

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

След като инсталацията завърши, е подходящо да се провери дали услугата Neutron работи:

openstack network agent list

Инсталиране на Cinder услуга

Създаване на базата данни на Cinder:

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';

Създаване на потребителя на Cinder и добавяне на роли:

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

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

Създаване на обекти за услуги на Cinder:

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

Създаваме точките за достъп на Cinder:

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

Инсталиране на Cinder пакети:

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

Редактираме /etc/cinder/cinder.conf, за да се конфигурира достъпа до базата данни, достъпа до опашката за съобщения на RabbitMQ и други необходими настройки:

[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

Попълваме базата данни на Cinder:

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

Следва да се рестартират услугите на Cinder:

sudo service cinder-scheduler restart

sudo service cinder-api restart

Проверяваме коректността на инсталацията:

openstack volume service list

Следващото нещо, което трябва да се направи, е да се настрои LVM. Първо, проверяваме дали имаме инсталиран lvm2:

sudo apt-get install lvm2

Ще е необходим физически обем за LVM. Това може да бъде дял, цял диск или RAID масив. Тук условно приемаме, че се нарича sdX.

sudo pvcreate /dev/sdX

Cinder ще използва LVM volume група, за да осигури логически volume за блоково съхранение. Създаване на група става със следната команда:

sudo vgcreate cinder-volumes /dev/sdX

Необходимо е да проверим дали е инсталиран инструментът за целево потребителско пространство iSCSI (tgtadm), който се използва от драйвера на LVM за експортиране на блокови устройства през iSCSI.

sudo apt-get install tgt

След като се промени конфигурацията, трябва да се рестартира cinder-volume service за прилагане на промените:

sudo systemctl restart openstack-cinder-volume

Инсталиране на Horizon

Инсталирането на Horizon, таблото за управление на OpenStack, включва няколко стъпки за настройване на средата, конфигуриране на необходимите компоненти и гарантиране, че тя се интегрира правилно със съществуващите услуги на OpenStack.

Инсталиране на пакета "Horizon":

sudo apt install openstack-dashboard

Първо, трябва да се конфигурират настройките на Horizon в /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"
```

Презареждане на уеб сървъра:

```
sudo systemctl reload apache2
```

Инсталиране на Sahara

Създаване на базата данни и потребителя на базата данни в 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;
```

Инсталиране на пакетите от Sahara:

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

Необходимо е да се редактира файла /etc/sahara/sahara.conf със следните директиви:

[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

Синхронизиране на базата данни:

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

Регистриране на Sahara в 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

Създаване на крайните точки на API за услуги в Sahara:

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

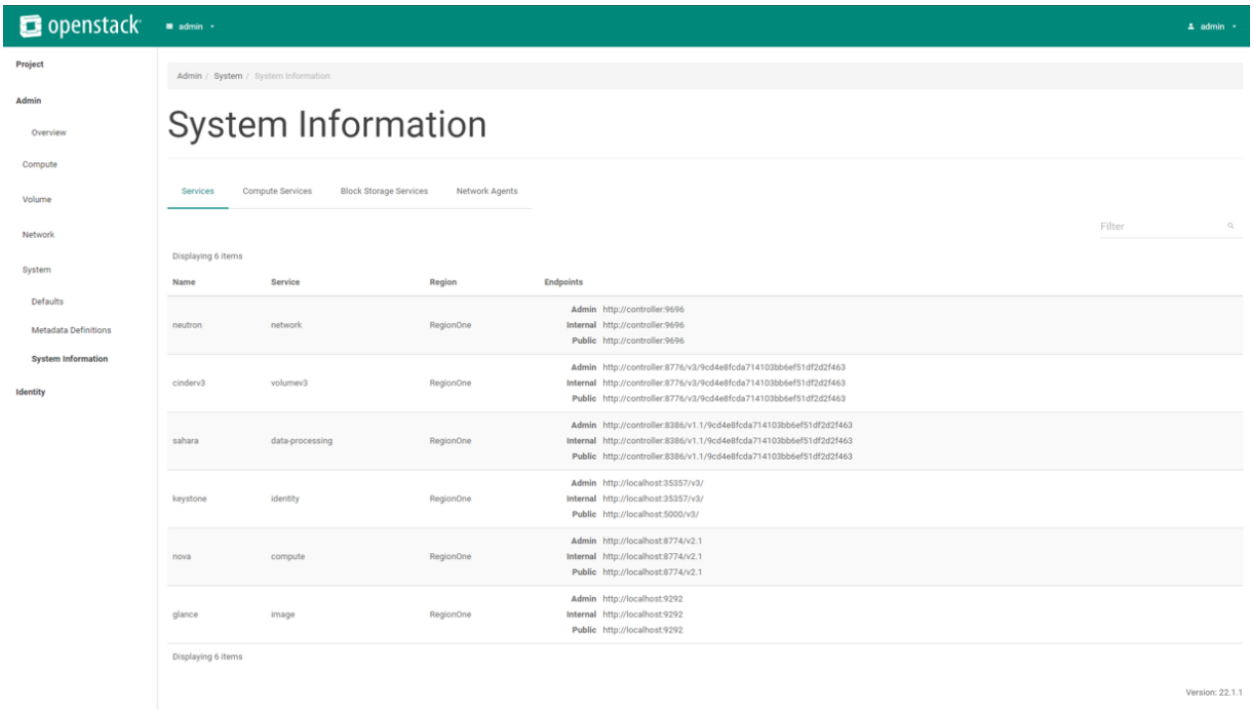
openstack endpoint create --region RegionOne data-processing internal http://controller:8386/v1.1/%\(tenant_id\)s

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

Накрая трябва да се рестартират услугите в Sahara:

service sahara-engine restart

При успешна инсталация и конфигурация на всички услуги, в Horizon може да се видят работещите услуги в текущата инсталация на OpenStack:



Литература

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