

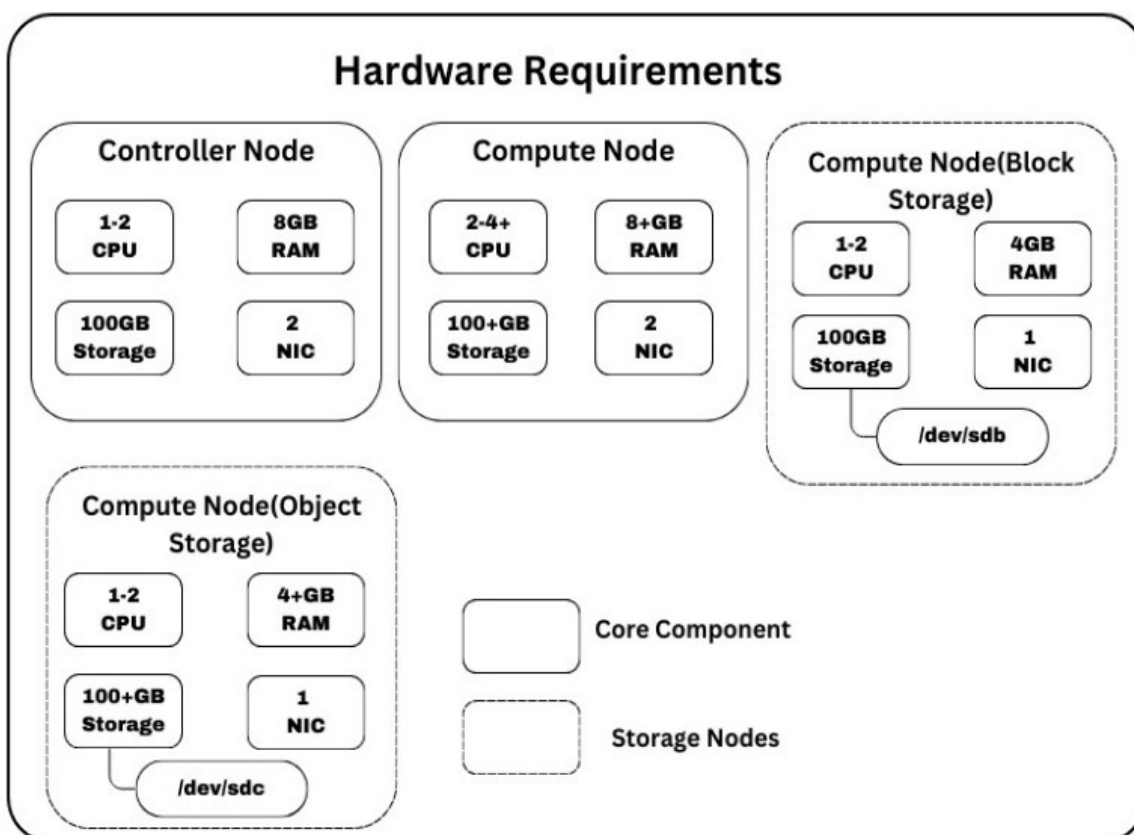
Openstack Setup

The OpenStack project is an open-source cloud computing platform for all types of clouds, which aims to be simple to implement, massively scalable, and feature-rich. Developers and cloud computing technologists from around the world created the OpenStack project.

OpenStack provides an Infrastructure-as-a-Service (IaaS) solution through interrelated services. Each service offers an Application Programming Interface (API) that facilitates this integration. Depending on your needs, you can install some or all services.

OpenStack is supported by a wide community of organisations, like RedHat, IBM, Canonical, Rackspace and Intel.

Hardware Requirements:



Components:

OpenStack consists of several core components that work together to deliver cloud infrastructure. Each component focuses on a specific function, such as computing, storage, networking, and identity management. Here are the key components of OpenStack:

1. **Controller:**

The controller node runs the Identity service, Image service, Placement service, management portions of Compute, management portion of Networking, various Networking agents, and the Dashboard. It also includes supporting services such as an SQL database, message queue, and NTP.

Optionally, the controller node runs portions of the Block Storage, Object Storage, Orchestration, and Telemetry services.

The controller node requires a minimum of two network interfaces.

2. Compute:

The compute node runs the hypervisor portion of Compute that operates instances. By default, Compute uses the KVM hypervisor. The compute node also runs a Networking service agent that connects instances to virtual networks and provides firewalling services to instances via security groups.

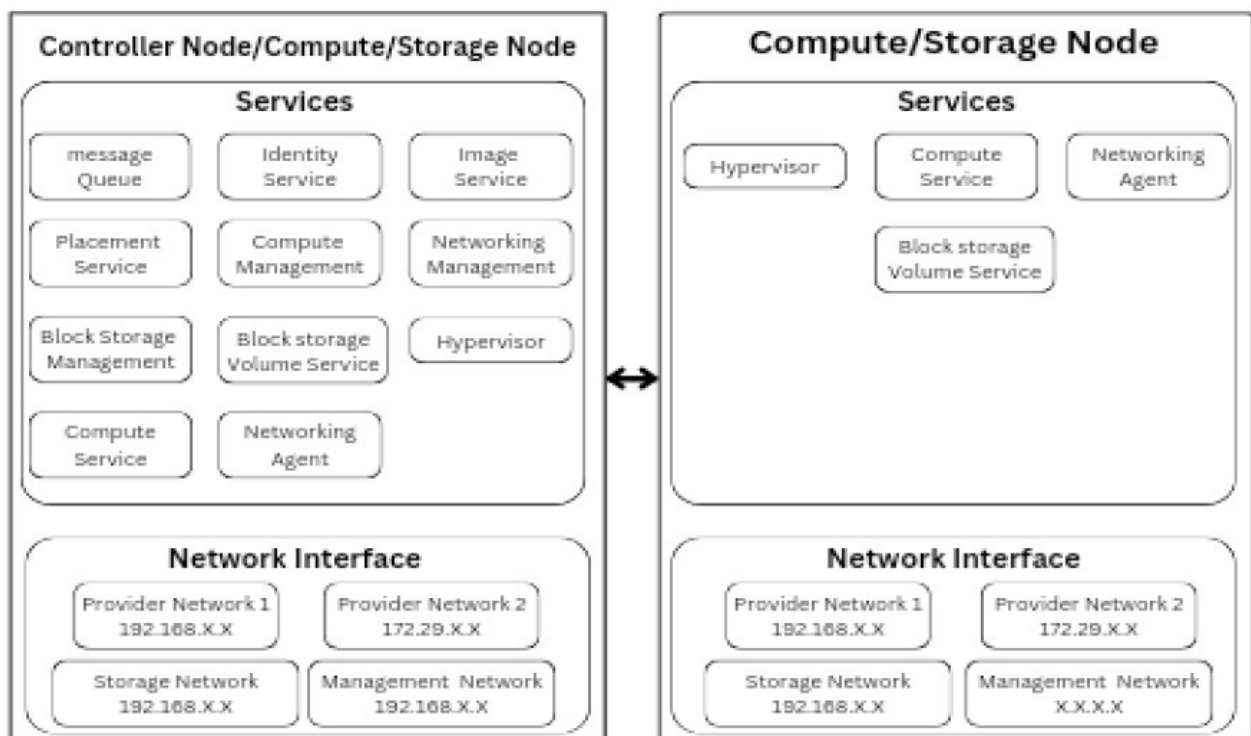
You can deploy more than one compute node. Each node requires a minimum of two network interfaces.

3. Block Storage:

The Block Storage node contains the disks that the Block Storage and Shared File System services provision for instances. But can deploy the block storage on the compute node also. Can deploy more than one block storage node. Each node requires a minimum of one network interface.

Basic Architecture for production:

OpenStack will be installed on a single system and also form clusters. But here, OpenStack implemented a multi-node installation architecture, in which a minimum of two nodes will be required. One node acts as a Controller Node and also act as a Compute and Storage Node, Second node acts as a Compute Node which will also act as a Storage Node in this OpenStack environment.



Services:

1. Keystone:

Keystone is one of the major projects in the OpenStack software stack. Keystone provides Identify, Token, Credential, Catalog and Policy related to OpenStack. Keystone performs user management and service catalog. User management consists user's permission and tracking while the service catalog consists availability of services with their API endpoints. All the installation was done at Controller Node.

2. Placement:

The placement service provides an HTTP API used to track resource provider inventories and usages. Placement is required by some of the other OpenStack services, notably Nova, therefore it should be installed before those other services but after Identity (keystone) service. All the installation was done at Controller Node.

3. Glance:

Glance image services include discovering, registering, and retrieving virtual machine (VM) images. It provides the ability to copy or snapshot a server image, and immediately store it. You can use stored images as templates to commission new servers quickly and more consistently than installing a server operating system and individually configuring services. All the installation was done at Controller Node.

4. Horizon:

Horizon is the canonical implementation of OpenStack's Dashboard, which provides a web-based user interface to OpenStack services including Nova, Swift, Keystone, etc. All the installation was done at Controller Node.

5. Nova:

Nova is the OpenStack project that provides a way to provision compute instances (aka virtual servers). Nova runs as a set of daemons on top of existing Linux servers to provide that service.

6. Neutron:

Neutron is an OpenStack project to provide "Network connectivity as a Service (NaaS)". OpenStack Networking handles the creation and management of a virtual networking infrastructure, including networks, switches, subnets, and routers for devices managed by the OpenStack Compute service (nova).

7. Cinder:

Cinder is the open-source OpenStack block storage software created to provide volumes to OpenStack Compute (Nova) virtual machines, or instances.

Networking:

1. Networking Option 1: Provider networks

The provider networks option deploys the OpenStack Networking service in the simplest way possible with primarily layer-2 (bridging/switching) services and VLAN segmentation of networks. Essentially, it bridges virtual networks to physical networks and relies on physical network infrastructure for layer-3 (routing) services and also works as a Public network.

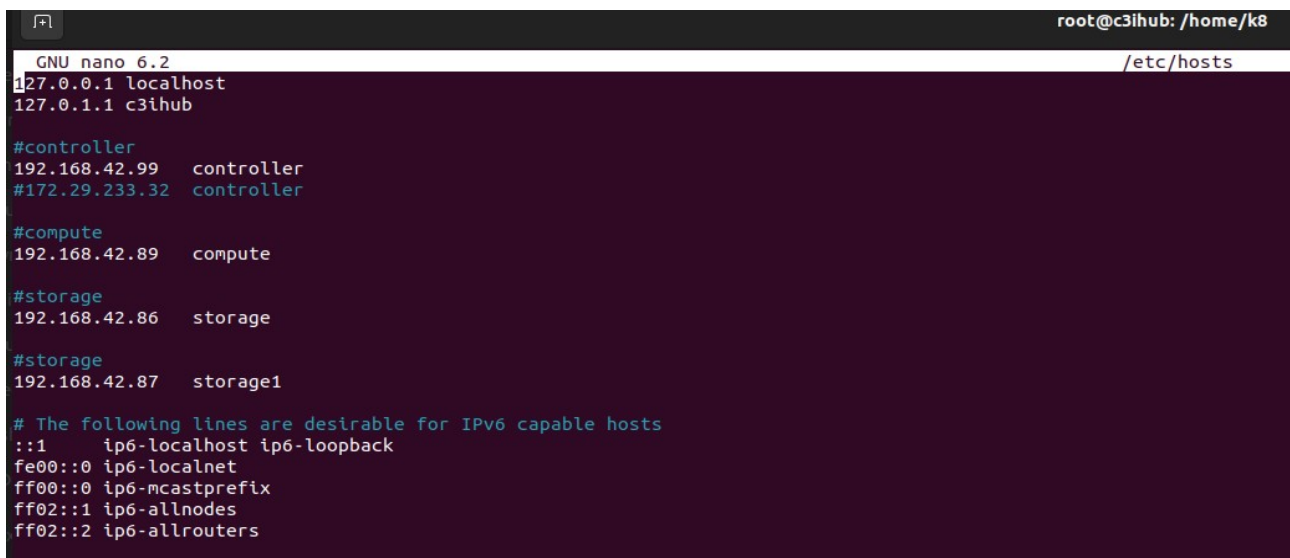
2. Networking Option 2: Self-service networks

The self-service networks option augments the provider networks option with layer-3 (routing) services that enable self-service networks using overlay segmentation methods such as VXLAN. Essentially, it routes virtual networks to physical networks using NAT. Additionally, this option provides the foundation for advanced services such as LBaaS and FWaaS.

Openstack Installation Steps for Controller Node/Compute Node/Storage Node

a) Configure name resolution

- Set the hostname of the node to controller.
- Edit the /etc/hosts file to contain the following:



```
root@c3ihub: /home/k8
GNU nano 6.2 /etc/hosts
127.0.0.1 localhost
127.0.1.1 c3ihub

#controller
192.168.42.99 controller
#172.29.233.32 controller

#compute
192.168.42.89 compute

#storage
192.168.42.86 storage

#storage
192.168.42.87 storage1

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Cli Login Script: save it as **admin-openrc** file.

```
export OS_USERNAME=admin
export OS_PASSWORD=A#45D875M*i#4n
export OS_PROJECT_NAME=admin
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_DOMAIN_NAME=Default
export OS_AUTH_URL=http://controller:5000/v3
export OS_IDENTITY_API_VERSION=3
```

b) Prerequisites Installation

Script download and setup the Chrony, as implementation of NTP, Openstack Package, Sql Database, Message queue, memcached and Etcd Datanase.

```
#!/bin/bash

# update the system
sudo apt update

# Define variables
NTP_SERVER="your.ntp.server" # Replace with your NTP server
SUBNET="10.0.2.15/24" # Replace with your subnet if different
MYSQL_BIND_ADDRESS="10.0.2.15" # Replace with your MySQL bind address
MEMCACHED_IP="10.0.2.15" # Replace with your management IP address for memcached
ETCD_IP="10.0.2.15" # Replace with your etcd management IP address

#---chrony---
# install the chrony
sudo apt install chrony -y

# Update chrony configuration
CONFIG_FILE="/etc/chrony/chrony.conf"
echo "Creating the backup of the original file"
cp $CONFIG_FILE $CONFIG_FILE.bak

echo "Updating /etc/chrony/chrony.conf..."
sudo bash -c "cat <<EOF >> $CONFIG_FILE
# Use public servers from the pool.ntp.org project.
server $NTP_SERVER iburst

# Allow clients from the local subnet
allow $SUBNET

# If necessary, replace $SUBNET with a description of your subnet.
EOF"

# Restart chrony service
echo "Restarting chrony service..."
sudo systemctl restart chrony

# verify Operation
echo "Verify Operations"
chronyc sources

#---MariaDB-----
# install MariaDB
sudo apt install mariadb-server python3-pymysql -y

# Configure MariaDB
echo "Updating /etc/mysql/mariadb.conf.d/99-openstack.cnf..."
sudo bash -c "cat <<EOF > /etc/mysql/mariadb.conf.d/99-openstack.cnf
[mysqld]
bind-address = $MYSQL_BIND_ADDRESS"
```

```

default-storage-engine = innodb
innodb_file_per_table = on
max_connections = 4096
collation-server = utf8_general_ci
character-set-server = utf8
EOF"

# Restart MySQL/MariaDB service
echo "Restarting MySQL/MariaDB service..."
sudo systemctl restart mysql

# -----Rabbit message queue-----
# Rabbit message queue installation and configuration
sudo apt install rabbitmq-server -y

# Add the openstack user and Replace RABBIT_PASS with a suitable password.
rabbitmqctl add_user openstack R123

# Permit configuration, write, and read access for the openstack user:
rabbitmqctl set_permissions openstack ".*" ".*" ".*"

#-----memcached Server-----
# install memcached server
sudo apt install memcached python3-memcache -y

# Configure memcached
CONFIG_FILE_mem="/etc/memcached.conf"
echo "Create the backup of the memcached.conf file"
cp $CONFIG_FILE_mem $CONFIG_FILE_mem.bak

echo "Updating /etc/memcached.conf..."
sudo bash -c "sed -i 's/-l .* -l $MEMCACHED_IP/ $CONFIG_FILE_mem'"

# Restart memcached service
echo "Restarting memcached service..."
sudo systemctl restart memcached

#-----etcd-----
# install etcd
sudo apt install etcd -y

# Configure etcd
CONFIG_FILE_etcd="/etc/default/etcd"
cp $CONFIG_FILE_etcd $CONFIG_FILE_etcd.bak

echo "Updating /etc/default/etcd..."
sudo bash -c "cat <<EOF >> $CONFIG_FILE_etcd
ETCD_NAME=\"controller\"
ETCD_DATA_DIR=\"/var/lib/etcd\"
ETCD_INITIAL_CLUSTER_STATE=\"new\"
ETCD_INITIAL_CLUSTER_TOKEN=\"etcd-cluster-01\"
ETCD_INITIAL_CLUSTER=\"controller=http://$ETCD_IP:2380\"
ETCD_INITIAL_ADVERTISE_PEER_URLS=\"http://$ETCD_IP:2380\"
ETCD_ADVERTISE_CLIENT_URLS=\"http://$ETCD_IP:2379\"
ETCD_LISTEN_PEER_URLS=\"http://0.0.0.0:2380\"
ETCD_LISTEN_CLIENT_URLS=\"http://$ETCD_IP:2379\"
EOF"

```

```
# Enable and restart etcd service
echo "Enabling and restarting etcd service..."
sudo systemctl enable etcd
sudo systemctl restart etcd

#-----openstack Package-----
#Openstack packages
sudo add-apt-repository cloud-archive:bobcat

#client installation
sudo apt install python3-openstackclient -y

echo "Openstack Prerequisites installation, Configuration and setup completed successfully!"
```

c) Keystone Installation

The OpenStack Identity service provides a single point of integration for managing authentication, authorization, and a catalog of services.

- **openssl rand -hex 10 #used to create password.**

```
#!/bin/bash

# Keystone installation

# Create Keystone database and grant privileges
DB_ROOT_PASS="123" # Replace with your MariaDB/MySQL root password
# Create the SQL commands to be executed
SQL_COMMANDS=$(cat <<EOF
CREATE DATABASE keystone;
GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' IDENTIFIED BY 'Pass';
GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' IDENTIFIED BY 'Pass';
EOF
)

# Run the SQL commands as root
echo "Setting up keystone databases and granting privileges..."

mysql -uroot -p"$DB_ROOT_PASS" -e "$SQL_COMMANDS"

if [ $? -eq 0 ]; then
    echo "Databases created and privileges granted successfully."
else
    echo "There was an error setting up the databases."
fi
echo "Database setup completed"

# install the package
sudo apt install keystone -y

#keystone configuration
config_file="/etc/keystone/keystone.conf"
# cp $config_file $config_file.bak
# Define the new connection string and token provider
NEW_DB_CONNECTION="connection = mysql+pymysql://keystone:password@controller/keystone"
NEW_TOKEN_PROVIDER="provider = fernet"
```

```

if [ -f "$config_file" ]; then
    echo "Updatng $config_file"
    sudo cp $config_file $config_file.bak

    # Check if the [database] section exists
    if grep -q "\[database\]" "$config_file"; then
        echo "[database] section found in $config_file"

        # Check if there is already a connection string in the [database] section
        if grep -A1 "\[database\]" "$config_file" | grep -q "^connection ="; then
            echo "Existing connection string found, commenting it out"

            # Comment out the existing connection string
            sudo sed -i "s/^.*connection = .*/# &/" "$config_file"
        else
            echo "No existing connection string found in the [database] section"
        fi

        # Add the new connection string below the [database] section
        sudo sed -i "/^\[database\]/a$NEW_DB_CONNECTION" "$config_file"
        echo "New connection string added to [database] section"

    else
        echo "[database] section not found in $config_file"
        echo "Adding the [database] section and new connection string"

        # If the [database] section doesn't exist, append it to the config file
        echo -e "\n[database]\n$NEW_DB_CONNECTION" | sudo tee -a "$config_file"
    fi

    # editing the token section
    if grep -q "\[token\]" "$config_file"; then
        echo "[token] section found in $config_file"

        # Check if there is already a connection string in the [database] section
        if grep -A1 "\[token\]" "$config_file" | grep -q "^provider ="; then
            echo "Existing provider string found, commenting it out"

            # Comment out the existing connection string
            sudo sed -i.bak "s/^.*provider = .*/# &/" "$config_file"
        else
            echo "No existing provider string found in the [token] section"
        fi

        # Add the new provider string below the [token] section
        sudo sed -i "/^\[token\]/a$NEW_TOKEN_PROVIDER" "$config_file"
        echo "New connection string added to [token] section"

    else
        echo "[token] section not found in $config_file"
        echo "Adding the [token] section and new connection string"

        # If the [token] section doesn't exist, append it to the config file
        echo -e "\n[token]\n$NEW_TOKEN_PROVIDER" | sudo tee -a "$config_file"
    fi

```



```

else
    echo "keystone configuration file $config_file not found. Exiting..."
    exit 1
fi
# Inform the user
echo "Keystone configuration updated successfully!"

#Populate the Identity service database:
su -s /bin/sh -c "keystone-manage db_sync" keystone
echo "database populated..."

#Initialize Fernet key repositories:
keystone-manage fernet_setup --keystone-user keystone --keystone-group keystone
keystone-manage credential_setup --keystone-user keystone --keystone-group keystone
echo "Keys initialized successfully"

#Bootstrap the Identity service:
keystone-manage bootstrap --bootstrap-password A123 \
    --bootstrap-admin-url http://controller:5000/v3/ \
    --bootstrap-internal-url http://controller:5000/v3/ \
    --bootstrap-public-url http://controller:5000/v3/ \
    --bootstrap-region-id RegionOne
echo "bootstrapping the identity service successfully."

conf="/etc/apache2/apache2.conf"
sudo bash -c "cat <<EOF >> $conf
ServerName controller"

echo "apache updated successfully"

# restart the service
service apache2 restart

cat <<EOF > /home/openstack/admin-openrc
export OS_USERNAME=admin
export OS_PASSWORD=A123
export OS_PROJECT_NAME=admin
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_DOMAIN_NAME=Default
export OS_AUTH_URL=http://controller:5000/v3
export OS_IDENTITY_API_VERSION=3
EOF
echo "admin-openrc created successfully"

chmod +x admin-openrc
. admin-openrc

```

d) Glance Installation

The Image service (glance) enables users to discover, register, and retrieve virtual machine images. It offers a REST API that enables you to query virtual machine image metadata and retrieve an actual image.

```
#!/bin/bash
#-----Glance Installation-----

DB_ROOT_PASS="123"

SQL_COMMANDS=$(cat <<EOF
CREATE DATABASE glance;
GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' IDENTIFIED BY 'g123';
GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%' IDENTIFIED BY 'g123';
EOF
)

mysql -uroot -p"$DB_ROOT_PASS" -e "$SQL_COMMANDS"

if [ $? -eq 0 ]; then
    echo "Databases created and privileges granted successfully."
else
    echo "There was an error setting up the databases."
fi
echo "Database setup completed"

. /home/openstack/admin-openrc

# create service project
openstack project create --domain default --description "Project description" service

# Create the Glance User and assign the admin role
openstack user create --domain default --password "g123" glance
openstack role add --project service --user glance admin
SERVICE_ID=$(openstack service create --name glance --description "OpenStack Image" image -f value -c id)
openstack role add --user glance --user-domain Default --system all reader

# Create Glance endpoints
openstack endpoint create --region RegionOne image public http://controller:9292
openstack endpoint create --region RegionOne image internal http://controller:9292
endpoint_id=$(openstack endpoint create --region RegionOne image admin http://controller:9292 -f value -c id)

# install the package
sudo apt install glance -y

# Create the OpenStack Service
#echo "Creating OpenStack service: $SERVICE_NAME..."
#SERVICE_ID=$(openstack service create --name "$SERVICE_NAME" --description
"$SERVICE_DESCRIPTION" "$SERVICE_TYPE" -f value -c id)
```

```

#configure the Glance Config file
config_file="/etc/glance/glance-api.conf"
NEW_DB_CONNECTION="connection = mysql+pymysql://glance:g123@controller/glance"
keystone_authtoken_config="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 = g123"
paste_deploy_conf="flavor = keystone"
glance_store_conf="stores = file,http"
default_store = file
filesystem_store_datadir = /var/lib/glance/images/"
oslo_limit_conf="auth_url = http://controller:5000"
auth_type = password
user_domain_id = default
username = $SERVICE_ID
system_scope = all
password = g123
endpoint_id = $endpoint_id
region_name = RegionOne"
default_conf="use_keystone_quotas = True"

if [ -f "$config_file" ]; then
    echo "Updatng $config_file"
    sudo cp $config_file $config_file.bak

    # Check if the [database] section exists
    if grep -q "^\[database\]" "$config_file"; then
        echo "[database] section found in $config_file"

        # Check if there is already a connection string in the [database] section
        if grep -A1 "^\[database\]" "$config_file" | grep -q "^connection ="; then
            echo "Existing connection string found, commenting it out"

            # Comment out the existing connection string
            sudo sed -i "s/^.*connection = .*/# &/" "$config_file"

        else
            echo "No existing connection string found in the [database] section"
        fi

        # Add the new connection string below the [database] section
        sudo sed -i "/^\[database\]/a$NEW_DB_CONNECTION" "$config_file"
        echo "New connection string added to [database] section"

    else
        echo "[database] section not found in $config_file"
        echo "Adding the [database] section and new connection string"

        # If the [database] section doesn't exist, append it to the config file
        echo -e "\n[database]\n$NEW_DB_CONNECTION" | sudo tee -a "$config_file"
    fi
fi

```

```

# Editing the [keystone_authtoken] section
if grep -q "\[keystone_authtoken\]" "$config_file"; then
    echo "[keystone_authtoken] section found in $config_file"

    # Add the keystone_authtoken below the [keystone_authtoken] section
    while IFS= read -r line; do
        key=$(echo "$line" | cut -d=' ' -f1 | xargs)
        value=$(echo "$line" | cut -d=' ' -f2- | xargs)

        # Check if the key exists in the section, if not, add it
        if grep -A10 "\[keystone_authtoken\]" "$config_file" | grep -q "^$key ="; then
            echo "Updating $key in keystone_authtoken section"
            sudo sed -i "\[keystone_authtoken\]/,/\[/ s/^$key =.*|$key = $value|" "$config_file"
        else
            echo "Adding $key to keystone_authtoken section"
            sudo sed -i "\[keystone_authtoken\]/a$key = $value" "$config_file"
        fi
    done <<< "$keystone_authtoken_config"
    echo "keystone_authtoken added to [keystone_authtoken] section"

else
    echo "[keystone_authtoken] section not found in $config_file"
    echo "Adding the [keystone_authtoken] section and new connection string"

    # If the [keystone_authtoken] section doesn't exist, append it to the config file
    #echo -e "\n[keystone_authtoken]\n$keystone_authtoken_config" | sudo tee -a "$config_file"
    echo "[keystone_authtoken]" >> "$config_file"

    # Add the keystone_authtoken below the [keystone_authtoken] section
    while IFS= read -r line; do
        key=$(echo "$line" | cut -d=' ' -f1 | xargs)
        value=$(echo "$line" | cut -d=' ' -f2- | xargs)
        # Check if the key exists in the section, if not, add it
        if grep -A10 "\[keystone_authtoken\]" "$config_file" | grep -q "^$key ="; then
            echo "Updating $key in keystone_authtoken section"
            sudo sed -i "\[keystone_authtoken\]/,/\[/ s/^$key =.*|$key = $value|" "$config_file"
        else
            echo "Adding $key to keystone_authtoken section"
            sudo sed -i "\[keystone_authtoken\]/a$key = $value" "$config_file"
        fi
    done <<< "$keystone_authtoken_config"
fi

# edit paste_deploy section
if grep -q "\[paste_deploy\]" "$config_file"; then
    echo "[paste_deploy] section found in $config_file"

    # Add the paste_deploy below the [paste_deploy] section
    sudo sed -i "\[paste_deploy\]/a$paste_deploy_conf" "$config_file"
    echo "paste_deploy added to [paste_deploy] section"

else
    echo "[paste_deploy] section not found in $config_file"
    echo "Adding the [paste_deploy] section and new connection string"

    # If the [paste_deploy] section doesn't exist, append it to the config file
    echo -e "\n[paste_deploy]\n$paste_deploy_conf" | sudo tee -a "$config_file"
fi

```

```

# edit glance_store section
if grep -q "^\[glance_store\]" "$config_file"; then
    echo "[glance_store] section found in $config_file"

    # Add the glance_store below the [glance_store] section
    while IFS= read -r line; do
        key=$(echo "$line" | cut -d=' ' -f1 | xargs)
        value=$(echo "$line" | cut -d=' ' -f2- | xargs)

        # Check if the key exists in the section, if not, add it
        if grep -A10 "^\[glance_store\]" "$config_file" | grep -q "^$key ="; then
            echo "Updating $key in glance_store section"
            sudo sed -i "/^\[glance_store\]/,/^\[ / s/^$key =.*|$key = $value|" "$config_file"
        else
            echo "Adding $key to glance_store section"
            sudo sed -i "/^\[glance_store\]/a$key = $value" "$config_file"
        fi
    done <<< "$glance_store_conf"
    echo "glance_store added to [glance_store] section"

else
    echo "[glance_store] section not found in $config_file"
    echo "Adding the [glance_store] section and new connection string"

    # If the [glance_store] section doesn't exist, append it to the config file
    #echo -e "\n[glance_store]\n$glance_store_conf" | sudo tee -a "$config_file"
    echo "[glance_store]" >> "$config_file"
    while IFS= read -r line; do
        key=$(echo "$line" | cut -d=' ' -f1 | xargs)
        value=$(echo "$line" | cut -d=' ' -f2- | xargs)
        # Check if the key exists in the section, if not, add it
        if grep -A10 "^\[glance_store\]" "$config_file" | grep -q "^$key ="; then
            echo "Updating $key in glance_store section"
            sudo sed -i "/^\[glance_store\]/,/^\[ / s/^$key =.*|$key = $value|" "$config_file"
        else
            echo "Adding $key to glance_store section"
            sudo sed -i "/^\[glance_store\]/a$key = $value" "$config_file"
        fi
    done <<< "$glance_store_conf"
fi

#edit the oslo_limit section
if grep -q "^\[oslo_limit\]" "$config_file"; then
    echo "[oslo_limit] section found in $config_file"

    if [ -z "$SERVICE_ID" ]; then
        echo "Failed to create the service. Exiting..."
        exit 1
    fi
    echo "Service created with ID: $SERVICE_ID"

    if [ -z "$Sendpoint_id" ]; then
        echo "Failed to create the service. Exiting..."
        exit 1
    fi
    echo "Service created with ID: $Sendpoint_id"

```

```

# Add the oslo_limit below the [oslo_limit] section
while IFS= read -r line; do
    key=$(echo "$line" | cut -d=' ' -f1 | xargs)
    value=$(echo "$line" | cut -d=' ' -f2- | xargs)

    # Check if the key exists in the section, if not, add it
    if grep -A10 "[oslo_limit]" "$config_file" | grep -q "^$key ="; then
        echo "Updating $key in oslo_limit section"
        sudo sed -i "/^[oslo_limit]/,/^[/ s|^$key =.*|$key = $value|" "$config_file"
    else
        echo "Adding $key to oslo_limit section"
        sudo sed -i "/^[oslo_limit]/a$key = $value" "$config_file"
    fi
done <<< "$oslo_limit_conf"
echo "oslo_limit added to [oslo_limit] section"

else
    echo "[oslo_limit] section not found in $config_file"
    echo "Adding the [oslo_limit] section and new connection string"

# If the [oslo_limit] section doesn't exist, append it to the config file
#echo -e "\n[oslo_limit]\n$oslo_limit_conf" | sudo tee -a "$config_file"
while IFS= read -r line; do
    key=$(echo "$line" | cut -d=' ' -f1 | xargs)
    value=$(echo "$line" | cut -d=' ' -f2- | xargs)
    # Check if the key exists in the section, if not, add it
    if grep -A10 "[oslo_limit]" "$config_file" | grep -q "^$key ="; then
        echo "Updating $key in oslo_limit section"
        sudo sed -i "/^[oslo_limit]/,/^[/ s|^$key =.*|$key = $value|" "$config_file"
    else
        echo "Adding $key to oslo_limit section"
        sudo sed -i "/^[oslo_limit]/a$key = $value" "$config_file"
    fi
done <<< "$oslo_limit_conf"
fi
# edit default section
if grep -q "[DEFAULT]" "$config_file"; then
    echo "[DEFAULT] section found in $config_file"

    # Add the paste_deploy below the [paste_deploy] section
    sudo sed -i "/^[DEFAULT]/a$default_conf" "$config_file"
    echo "default configuration added to [DEFAULT] section"
else
    echo "[DEFAULT] section not found in $config_file"
    echo "Adding the [paste_deploy] section and new connection string"
# If the [DEFAULT] section doesn't exist, append it to the config file
echo -e "\n[DEFAULT]\n$default_conf" | sudo tee -a "$config_file"
fi
else
    echo "glance configuration file $config_file not found. Exiting..."
    exit 1
fi
# Populate the Image service database:
su -s /bin/sh -c "glance-manage db_sync" glance
# Restart the Image services:
sudo service glance-api restart

```

e) Placement Installation:

The placement service tracks the inventory and usage of each provider. For example, an instance created on a compute node may be a consumer of resources such as RAM and CPU from a compute node resource provider, disk from an external shared storage pool resource provider and IP addresses from an external IP pool resource provider.

```
#!/bin/bash

#----placement Service configuration and installation----

DB_ROOT_PASS="123"
SQL_COMMANDS=$( cat <<EOF
CREATE DATABASE placement;
GRANT ALL PRIVILEGES ON placement.* TO 'placement'@'localhost' \
  IDENTIFIED BY 'p123';
GRANT ALL PRIVILEGES ON placement.* TO 'placement'@'%'\
  IDENTIFIED BY 'p123';
EOF
)

mysql -uroot -p"$DB_ROOT_PASS" -e "$SQL_COMMANDS"
if [ $? -eq 0 ]; then
  echo "Placement database created and privileges granted successfully"
else
  echo "There was an error setting up the database"
fi
echo "database setup complete"

. /home/openstack/admin-openrc

# Create a Placement service user
openstack user create --domain default --password "p123" placement

# Add the Placement user to the service project with the admin role
openstack role add --project service --user placement admin

# Create the Placement API entry in the service catalog:
openstack service create --name placement --description "Placement API" placement

# Create the Placement API service endpoints
openstack endpoint create --region RegionOne placement public http://controller:8778
openstack endpoint create --region RegionOne placement internal http://controller:8778
openstack endpoint create --region RegionOne placement admin http://controller:8778

# Install the packages:
sudo apt install placement-api -y

# Edit the /etc/placement/placement.conf file and complete the following actions
#configure the placement Config file
config_file="/etc/placement/placement.conf"
NEW_DB_CONNECTION="connection = mysql+pymysql://placement:p123@controller/placement"
keystone_authtoken_config="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 = p123"
api_conf="auth_strategy = keystone"

# check the placement configuration file exists or not
if [ -f "$config_file" ]; then
    echo "Updating $config_file"
    sudo cp $config_file $config_file.bak
    echo "backup created for the file"

    # Check if the [placement_database] section exists
    if grep -q "\[placement_database\]" "$config_file"; then
        echo "[placement_database] section found in $config_file"

        # Check if there is already a connection string in the [placement_database] section
        if grep -A1 "\[placement_database\]" "$config_file" | grep -q "^connection ="; then
            echo "Existing connection string found, commenting it out"

            # Comment out the existing connection string
            sudo sed -i "s/^.*connection = .*/# &/" "$config_file"

        else
            echo "No existing connection string found in the [placement_database] section"
        fi

        # Add the new connection string below the [placement_database] section
        sudo sed -i "/^\[placement_database\]/a$NEW_DB_CONNECTION" "$config_file"
        echo "New connection string added to [placement_database] section"

    else
        echo "[placement_database] section not found in $config_file"
        echo "Adding the [placement_database] section and new connection string"

        # If the [placement_database] section doesn't exist, append it to the config file
        echo -e "\n[placement_database]\n$NEW_DB_CONNECTION" | sudo tee -a "$config_file"
    fi

    # Editing the [keystone_authtoken] section
    if grep -q "\[keystone_authtoken\]" "$config_file"; then
        echo "[keystone_authtoken] section found in $config_file"

        # Add the keystone_authtoken below the [keystone_authtoken] section
        while IFS= read -r line; do
            key=$(echo "$line" | cut -d=' ' -f1 | xargs)
            value=$(echo "$line" | cut -d=' ' -f2- | xargs)

            # Check if the key exists in the section, if not, add it
            if grep -A10 "\[keystone_authtoken\]" "$config_file" | grep -q "^$key ="; then
                echo "Updating $key in keystone_authtoken section"
                sudo sed -i "/^\[keystone_authtoken\]/,/^[/ s/^$key =.*|$key = $value|" "$config_file"
            else
                echo "Adding $key to keystone_authtoken section"
                sudo sed -i "/^\[keystone_authtoken\]/a$key = $value" "$config_file"
            fi
        done
    fi

```



```

done <<< "$keystone_authtoken_config"
echo "keystone_authtoken added to [keystone_authtoken] section"

else
    echo "[keystone_authtoken] section not found in $config_file"
    echo "Adding the [keystone_authtoken] section and new connection string"
    # If the [keystone_authtoken] section doesn't exist, append it to the config file
    sudo bash -c "cat <<EOF >> $config_file
[keystone_authtoken]
EOF"
    # Add the keystone_authtoken below the [keystone_authtoken] section
    while IFS= read -r line; do
        key=$(echo "$line" | cut -d'=' -f1 | xargs)
        value=$(echo "$line" | cut -d'=' -f2- | xargs)
        # Check if the key exists in the section, if not, add it
        if grep -A10 "\[keystone_authtoken\]" "$config_file" | grep -q "^$key ="; then
            echo "Updating $key in keystone_authtoken section"
            sudo sed -i "/\[keystone_authtoken\]/,/\[/ s/^$key =.*|$key = $value|" "$config_file"
        else
            echo "Adding $key to keystone_authtoken section"
            sudo sed -i "/\[keystone_authtoken\]/a$key = $value" "$config_file"
        fi
    done <<< "$keystone_authtoken_config"
fi
# edit api section
if grep -q "\[api\]" "$config_file"; then
    echo "[api] section found in $config_file"
    # Add the api below the [api] section
    sudo sed -i "/\[api\]/a$api_conf" "$config_file"
    echo "paste_deploy added to [api] section"

else
    echo "[api] section not found in $config_file"
    echo "Adding the [api] section and new connection string"
    # If the [api] section doesn't exist, append it to the config file
    echo -e "\n[api]\n$api_conf" | sudo tee -a "$config_file"
fi

else
    echo "glance configuration file $config_file not found. Exiting..."
    exit 1
fi

# Populate the placement database:
su -s /bin/sh -c "placement-manage db sync" placement

# Reload the web server
sudo service apache2 restart

# verify the installation
#. admin-openrc

# Perform status checks to make sure everything is in order:
placement-status upgrade check

# Run some commands against the placement API and Install the osc-placement plugin.
pip3 install osc-placement

```

f) Horizon Installation

The only core service required by the dashboard is the Identity service.

```
#!/bin/bash

#---create Horizon dashboard-----
#Install the packages
sudo apt install openstack-dashboard -y

# Configure the Openstack dashboard configuration file
config_file="/etc/openstack-dashboard/local_settings.py"

sudo cp $config_file $config_file.bak
echo "backup completed successfully"

# Configure the dashboard to use OpenStack services on the controller node.
sudo sed -i "s/^.*OPENSTACK_HOST = .*/# &/" "$config_file"
echo 'OPENSTACK_HOST = "controller" ' >> $config_file

#In the Dashboard configuration section, allow your hosts to access Dashboard.
sudo sed -i "s/^.*ALLOWED_HOSTS = .*/# &/" "$config_file"
echo 'ALLOWED_HOSTS = ["*"] ' >> "$config_file"

#Configure the memcached session storage service:
sudo sed -i "s/^.*SESSION_ENGINE = .*/# &/" "$config_file"
echo "SESSION_ENGINE = 'django.contrib.sessions.backends.cache'" >> $config_file

NEW_BACKEND="django.core.cache.backends.memcached.MemcachedCache"
NEW_LOCATION="controller:11211"

# Check if the 'CACHES' section exists
if grep -q "CACHES" "$config_file"; then
    echo "CACHES section found in $config_file"

    # Update the BACKEND value in the CACHES section
    if grep -q "'BACKEND':" "$config_file"; then
        echo "Updating BACKEND value"
        sudo sed -i "s/'BACKEND':.*/'BACKEND': '$NEW_BACKEND',/" "$config_file"
    else
        echo "BACKEND key not found, adding it"
        sudo sed -i "/CACHES = {/,/}/ s/^(default:.*)\\n\\n\\n'BACKEND': '$NEW_BACKEND',/"
"$config_file"
    fi

    # Update the LOCATION value in the CACHES section
    if grep -q "'LOCATION':" "$config_file"; then
        echo "Updating LOCATION value"
        sudo sed -i "s/'LOCATION':.*/'LOCATION': '$NEW_LOCATION',/" "$config_file"
    else
        echo "LOCATION key not found, adding it"
        sudo sed -i "/CACHES = {/,/}/ s/^(default:.*)\\n\\n\\n'LOCATION': '$NEW_LOCATION',/"
"$config_file"
    fi
else
    echo "CACHES section not found in $config_file"
fi
```

```

echo "Cache configuration update complete."
# Enable the Identity API version
sudo sed -i "s/^.*OPENSTACK_KEYSTONE_URL = .*/# &/" "$config_file"
echo 'OPENSTACK_KEYSTONE_URL = "http://s:5000/v3" % OPENSTACK_HOST' >> $config_file

# configure the time zone
sudo sed -i "s/^.*TIME_ZONE = .*/# &/" "$config_file"
echo 'TIME_ZONE = "Asia/Kolkata"' >> $config_file

# Configure theme
sudo sed -i "s/^.*DEFAULT_THEME = .*/# &/" "$config_file"
echo "DEFAULT_THEME = 'ubuntu' " >> $config_file

# configure the webroot
sudo sed -i "s/^.*WEBROOT = .*/# &/" "$config_file"
echo "WEBROOT='/horizon/'" >> $config_file

# Enable support for domains:
echo "OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True" >> $config_file

sudo bash -c 'cat <<EOF >> $config_file
OPENSTACK_API_VERSIONS = {
    "identity": 3,
    "image": 2,
    "volume": 2,
}

OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = "Default"
OPENSTACK_KEYSTONE_DEFAULT_ROLE = "user"

OPENSTACK_NEUTRON_NETWORK = {
    #...
    "enable_router": True,
    "enable_quotas": True,
    "enable_ipv6": True,
    "enable_distributed_router": True,
    "enable_ha_router": True,
    "enable_fip_topology_check": True,
}
EOF'

sudo chown horizon:horizon /usr/lib/python3/dist-packages/openstack_dashboard/local/local_settings.py
sudo chown -R horizon:horizon /var/lib/openstack-dashboard/
sudo chmod -R 755 /var/lib/openstack-dashboard/
sudo chmod -R 600 /var/lib/openstack-dashboard/secret_key
sudo chown -R horizon:horizon /usr/share/openstack-dashboard/
sudo chmod -R 755 /usr/share/openstack-dashboard/

#sudo su -s /bin/sh -c "python3 /usr/share/openstack-dashboard/manage.py collectstatic --noinput" horizon

conf_file1="/etc/apache2/conf-available/openstack-dashboard.conf"
sudo bash -c "cat <<EOF >> $conf_file1
WSGIApplicationGroup %{GLOBAL}
EOF"

#Reload the web server configuration:
systemctl reload apache2.service

```

g) Nova (Compute Service) Installation

OpenStack Compute to host and manage cloud computing systems. OpenStack Compute is a major part of an Infrastructure-as-a-Service (IaaS) system. The main modules are implemented in Python.

This script contains the controller node as controller node as well as compute node. So it will install the hypervisor as well.

```
#!/bin/bash

#---- Nova Installation-----

DB_ROOT_PASS="123"
SQL_COMMANDS=$(cat <<EOF
CREATE DATABASE nova_api;
CREATE DATABASE nova;
CREATE DATABASE nova_cell0;
GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'localhost' IDENTIFIED BY 'no123';
GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'%' IDENTIFIED BY 'no123';
GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' IDENTIFIED BY 'no123';
GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' IDENTIFIED BY 'no123';

GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova'@'localhost' IDENTIFIED BY 'no123';
GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova'@'%' IDENTIFIED BY 'no123';
EOF
)
mysql -uroot -p"$DB_ROOT_PASS" -e "$SQL_COMMANDS"

if [ $? -eq 0 ]; then
    echo "Databases created and privileges granted successfully."
else
    echo "There was an error setting up the databases."
fi
echo "Database setup completed"

. /home/openstack/admin-openrc
# create nova user and assign the admin role to nova user
openstack user create --domain default --password "no123" nova
openstack role add --project service --user nova admin

# create the nova user entity
openstack service create --name nova --description "OpenStack Compute" compute

# Create the Compute API service endpoints
openstack endpoint create --region RegionOne compute public http://controller:8774/v2.1
openstack endpoint create --region RegionOne compute internal http://controller:8774/v2.1
openstack endpoint create --region RegionOne compute admin http://controller:8774/v2.1

# Install the packages:
sudo apt install nova-api nova-conductor nova-novncproxy nova-scheduler nova-compute -y

# dit the /etc/nova/nova.conf file and complete the following actions. In the [api_database] and [database]
sections, configure database access.

metadata_id=$(openssl rand -hex 10)
```

```

config_file="/etc/nova/nova.conf"
api_db="connection = mysql+pymysql://nova:no123@controller/nova_api"
new_db="connection = mysql+pymysql://nova:no123@controller/nova"
trans_url="transport_url = rabbit://openstack:R123@controller:5672/"
api="auth_strategy = keystone"
keystone_authtoken_config="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 = nova
password = no123"
service_user="send_service_user_token = true
auth_url = http://controller:5000/identity
auth_strategy = keystone
auth_type = password
project_domain_name = Default
project_name = service
user_domain_name = Default
username = nova
password = no123"
my_ip="my_ip=10.0.2.15"
vnc="enabled = true
server_listen = 0.0.0.0
server_proxyclient_address = 10.0.2.15
novncproxy_base_url = http://172.29.24.217:6080/vnc_auto.html"
glance="api_servers = http://controller:9292"
oslo_con="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 = p123"
neutron="region_name = RegionOne
project_domain_name = Default
project_name = service
auth_type = password
user_domain_name = Default
auth_url = http://controller:5000/v3
username = neutron
password = ne123
service_metadata_proxy = true
metadata_proxy_shared_secret = $metadata_id"
scheduler="discover_hosts_in_cells_interval = 300"
if [ -f "$config_file" ]; then
    echo "updating the $config_file"
    sudo cp $config_file $config_file.bak

    #In the [api_database] and [database] sections, configure database access.
    if grep -q "^\[api_database\]" "$config_file"; then
        echo "[api_database] section found in $config_file"
    fi
fi

```

```

# Check if there is already a connection string in the [api_database] section
if grep -A1 "^\[api_database\]" "$config_file" | grep -q "^connection ="; then
    echo "Existing connection string found, commenting it out"

    # Comment out the existing connection string
    sudo sed -i "s/^\.*connection = .*/# &/" "$config_file"

else
    echo "No existing connection string found in the [api_database] section"
fi

# Add the new connection string below the [api_database] section
sudo sed -i "/^\[api_database\]/a$api_db" "$config_file"
echo "New connection string added to [api_database] section"

else
    echo "[api_database] section not found in $config_file"
    echo "Adding the [api_database] section and new connection string"

# If the [api_database] section doesn't exist, append it to the config file
echo -e "\n\[api_database\]\n$api_db" | sudo tee -a "$config_file"
fi

# Check if the [database] section exists
if grep -q "^\[database\]" "$config_file"; then
    echo "[database] section found in $config_file"

    # Check if there is already a connection string in the [database] section
    if grep -A1 "^\[database\]" "$config_file" | grep -q "^connection ="; then
        echo "Existing connection string found, commenting it out"

        # Comment out the existing connection string
        sudo sed -i "s/^\.*connection = .*/# &/" "$config_file"

    else
        echo "No existing connection string found in the [database] section"
    fi

    # Add the new connection string below the [database] section
    sudo sed -i "/^\[database\]/a$new_db" "$config_file"
    echo "New connection string added to [database] section"

else
    echo "[database] section not found in $config_file"
    echo "Adding the [database] section and new connection string"

# If the [database] section doesn't exist, append it to the config file
echo -e "\n\[database\]\n$new_db" | sudo tee -a "$config_file"
fi

# Check if the [DEFAULT] section exists
if grep -q "^\[DEFAULT\]" "$config_file"; then
    echo "[DEFAULT] section found in $config_file"

    #sudo sed -i "/^\[DEFAULT\]/a$my_ip" "$config_file"
    #echo "my_ip added to [DEFAULT] section"

```

```

# Check if there is already a transport_url string in the [Default] section
if grep -A1 "^\[DEFAULT\]" "$config_file" | grep -q "^transport_url ="; then
    echo "Existing transport_url string found, commenting it out"

    # Comment out the existing transport_url string
    sudo sed -i "s/^\.*transport_url = .*/# &/" "$config_file"

else
    echo "No existing connection string found in the [default] section"
fi

# Add the new connection string below the [database] section
sudo sed -i "/^\[DEFAULT\]/a$trans_url" "$config_file"
echo "New connection string added to [DEFAULT] section"

else
    echo "[DEFAULT] section not found in $config_file"
    echo "Adding the [DEFAULT] section and new connection string"

# If the [DEFAULT] section doesn't exist, append it to the config file
echo -e "\n[DEFAULT]\n\n$trans_url" | sudo tee -a "$config_file"
fi

# Check if the [DEFAULT] section exists
if grep -q "^\[DEFAULT\]" "$config_file"; then
    echo "[DEFAULT] section found in $config_file"

    sudo sed -i "/^\[DEFAULT\]/a$my_ip" "$config_file"
    echo "my_ip added to [DEFAULT] section"
else
    echo "[DEFAULT] section not found in $config_file"
    echo "Adding the [DEFAULT] section and new connection string"

# If the [DEFAULT] section doesn't exist, append it to the config file
echo -e "\n[DEFAULT]\n\n$my_ip" | sudo tee -a "$config_file"
fi

# edit the [api] section for keystone
if grep -q "^\[api\]" "$config_file"; then
    echo "[api] section found in $config_file"

    # Add the paste_deploy below the [paste_deploy] section
    sudo sed -i "/^\[api\]/a$api" "$config_file"
    echo "auth_strategy added to [api] section"

else
    echo "[api] section not found in $config_file"
    echo "Adding the [api] section and keystone auth"

# If the [api] section doesn't exist, append it to the config file
echo -e "\n[api]\n\n$api" | sudo tee -a "$config_file"
fi

# Editing the [keystone_authtoken] section
if grep -q "^\[keystone_authtoken\]" "$config_file"; then
    echo "[keystone_authtoken] section found in $config_file"

```

```

# Add the keystone_auth token below the [keystone_auth token] section
while IFS= read -r line; do
    key=$(echo "$line" | cut -d=' ' -f1 | xargs)
    value=$(echo "$line" | cut -d=' ' -f2- | xargs)

    # Check if the key exists in the section, if not, add it
    if grep -A10 "^\[keystone_auth token\]" "$config_file" | grep -q "^\$key ="; then
        echo "Updating $key in keystone_auth token section"
        sudo sed -i "/^\[keystone_auth token\]/,/^\[/ s/^\$key =.*|\$key = $value|" "$config_file"
    else
        echo "Adding $key to keystone_auth token section"
        sudo sed -i "/^\[keystone_auth token\]/a\$key = $value" "$config_file"
    fi
done <<< "$keystone_auth token_config"
echo "keystone_auth token added to [keystone_auth token] section"

else
    echo "[keystone_auth token] section not found in $config_file"
    echo "Adding the [keystone_auth token] section and new connection string"

# If the [keystone_auth token] section doesn't exist, append it to the config file
echo "[keystone_auth token]" >> "$config_file"

# Add the keystone_auth token below the [keystone_auth token] section
while IFS= read -r line; do
    key=$(echo "$line" | cut -d=' ' -f1 | xargs)
    value=$(echo "$line" | cut -d=' ' -f2- | xargs)
    # Check if the key exists in the section, if not, add it
    if grep -A10 "^\[keystone_auth token\]" "$config_file" | grep -q "^\$key ="; then
        echo "Updating $key in keystone_auth token section"
        sudo sed -i "/^\[keystone_auth token\]/,/^\[/ s/^\$key =.*|\$key = $value|" "$config_file"
    else
        echo "Adding $key to keystone_auth token section"
        sudo sed -i "/^\[keystone_auth token\]/a\$key = $value" "$config_file"
    fi
done <<< "$keystone_auth token_config"
fi

# Editing the [service_user] section
if grep -q "^\[service_user\]" "$config_file"; then
    echo "[service_user] section found in $config_file"

    # Add the service_user below the [service_user] section
    while IFS= read -r line; do
        key=$(echo "$line" | cut -d=' ' -f1 | xargs)
        value=$(echo "$line" | cut -d=' ' -f2- | xargs)

        # Check if the key exists in the section, if not, add it
        if grep -A10 "^\[service_user\]" "$config_file" | grep -q "^\$key ="; then
            echo "Updating $key in service_user section"
            sudo sed -i "/^\[service_user\]/,/^\[/ s/^\$key =.*|\$key = $value|" "$config_file"
        else
            echo "Adding $key to service_user section"
            sudo sed -i "/^\[service_user\]/a\$key = $value" "$config_file"
        fi
    done <<< "$service_user_config"
fi

```



```

done <<< "$service_user"
    echo "service_user added to [service_user] section"
else
    echo "[service_user] section not found in $config_file"
    echo "Adding the [service_user] section and new connection string"
    # If the [service_user] section doesn't exist, append it to the config file
    echo "[service_user]" >> "$config_file"
    # Add the service_user below the [service_user] section
    while IFS= read -r line; do
        key=$(echo "$line" | cut -d'=' -f1 | xargs)
        value=$(echo "$line" | cut -d'=' -f2- | xargs)
        # Check if the key exists in the section, if not, add it
        if grep -A10 "^\[service_user\]" "$config_file" | grep -q "^$key ="; then
            echo "Updating $key in service_user section"
            sudo sed -i "/^\[service_user\]/,/^\[/ s|^$key =.*|$key = $value|" "$config_file"
        else
            echo "Adding $key to service_user section"
            sudo sed -i "/^\[service_user\]/a$key = $value" "$config_file"
        fi
    done <<< "$service_user"
fi
# edit the [vnc] section
if grep -q "^\[vnc\]" "$config_file"; then
    echo "[vnc] section found in $config_file"
    # Add the vnc below the [vnc] section
    while IFS= read -r line; do
        key=$(echo "$line" | cut -d'=' -f1 | xargs)
        value=$(echo "$line" | cut -d'=' -f2- | xargs)
        # Check if the key exists in the section, if not, add it
        if grep -A10 "^\[vnc\]" "$config_file" | grep -q "^$key ="; then
            echo "Updating $key in vnc section"
            sudo sed -i "/^\[vnc\]/,/^\[/ s|^$key =.*|$key = $value|" "$config_file"
        else
            echo "Adding $key to vnc section"
            sudo sed -i "/^\[vnc\]/a$key = $value" "$config_file"
        fi
    done <<< "$vnc"
    echo "service_user added to [vnc] section"
else
    echo "[vnc] section not found in $config_file"
    echo "Adding the [vnc] section and new connection string"
    # If the [vnc] section doesn't exist, append it to the config file
    echo "[vnc]" >> "$config_file"
    # Add the vnc below the [vnc] section
    while IFS= read -r line; do
        key=$(echo "$line" | cut -d'=' -f1 | xargs)
        value=$(echo "$line" | cut -d'=' -f2- | xargs)
        # Check if the key exists in the section, if not, add it
        if grep -A10 "^\[vnc\]" "$config_file" | grep -q "^$key ="; then
            echo "Updating $key in vnc section"
            sudo sed -i "/^\[vnc\]/,/^\[/ s|^$key =.*|$key = $value|" "$config_file"
        else
            echo "Adding $key to vnc section"
            sudo sed -i "/^\[vnc\]/a$key = $value" "$config_file"
        fi
    done <<< "$vnc"
fi

```

```

# edit glance section
if grep -q "\[glance\]" "$config_file"; then
    echo "[glance] section found in $config_file"

    # Add the api_server below the [glance] section
    sudo sed -i "\[glance\]/a$glance" "$config_file"
    echo "glance added to [glance] section"

else
    echo "[glance] section not found in $config_file"
    echo "Adding the [glance] section and new connection string"

    # If the [glance] section doesn't exist, append it to the config file
    echo -e "\n[glance]\n$glance" | sudo tee -a "$config_file"
fi

# edit oslo_concurrency section
if grep -q "\[oslo_concurrency\]" "$config_file"; then
    echo "[oslo_concurrency] section found in $config_file"
    # Add the lock_path below the [oslo_concurrency] section
    sudo sed -i "\[oslo_concurrency\]/a$oslo_con" "$config_file"
    echo "lock_path added to [oslo_concurrency] section"

else
    echo "[oslo_concurrency] section not found in $config_file"
    echo "Adding the [oslo_concurrency] section and new connection string"

    # If the [oslo_concurrency] section doesn't exist, append it to the config file
    echo -e "\n[oslo_concurrency]\n$oslo_con" | sudo tee -a "$config_file"
fi

# edit the [placement] section
if grep -q "\[placement\]" "$config_file"; then
    echo "[placement] section found in $config_file"
    # Add the placement-conf below the [placement] section
    while IFS= read -r line; do
        key=$(echo "$line" | cut -d'=' -f1 | xargs)
        value=$(echo "$line" | cut -d'=' -f2- | xargs)

        # Check if the key exists in the section, if not, add it
        if grep -A10 "\[placement\]" "$config_file" | grep -q "^$key ="; then
            echo "Updating $key in placement section"
            sudo sed -i "\[placement\]/,/\[/ s/^$key =.*|$key = $value|" "$config_file"
        else
            echo "Adding $key to placement section"
            sudo sed -i "\[placement\]/a$key = $value" "$config_file"
        fi
    done <<< "$placement"
    echo "placement-conf added to [placement] section"

else
    echo "[placement] section not found in $config_file"
    echo "Adding the [placement] section and new connection string"

    # If the [placement] section doesn't exist, append it to the config file
    echo -e "\n[placement]\n$oslo_con" | sudo tee -a "$config_file"
fi

```

```

# edit the [neutron] section
if grep -q "^\[neutron\]" "$config_file"; then
    echo "[neutron] section found in $config_file"
    # Add the neutron-conf below the [neutron] section
    while IFS= read -r line; do
        key=$(echo "$line" | cut -d=' ' -f1 | xargs)
        value=$(echo "$line" | cut -d=' ' -f2- | xargs)
        # Check if the key exists in the section, if not, add it
        if grep -A10 "^\[neutron\]" "$config_file" | grep -q "^\$key ="; then
            echo "Updating $key in neutron section"
            sudo sed -i "/^\[neutron\]/,/^\[/ s/^\$key =.*|\$key = \$value|" "$config_file"
        else
            echo "Adding $key to neutron section"
            sudo sed -i "/^\[neutron\]/a\$key = \$value" "$config_file"
        fi
    done <<< "$neutron"
    echo "neutron-conf added to [neutron] section"
else
    echo "[neutron] section not found in $config_file"
    echo "Adding the [neutron] section and new connection string"
    # If the [neutron] section doesn't exist, append it to the config file
    echo -e "\n\[neutron\]\n\$oslo_con" | sudo tee -a "$config_file"
    # Add the neutron-conf below the [neutron] section
    while IFS= read -r line; do
        key=$(echo "$line" | cut -d=' ' -f1 | xargs)
        value=$(echo "$line" | cut -d=' ' -f2- | xargs)
        # Check if the key exists in the section, if not, add it
        if grep -A10 "^\[neutron\]" "$config_file" | grep -q "^\$key ="; then
            echo "Updating $key in neutron section"
            sudo sed -i "/^\[neutron\]/,/^\[/ s/^\$key =.*|\$key = \$value|" "$config_file"
        else
            echo "Adding $key to neutron section"
            sudo sed -i "/^\[neutron\]/a\$key = \$value" "$config_file"
        fi
    done <<< "$neutron"
    echo "neutron-conf added to [neutron] section"
fi
# edit scheduler section
if grep -q "^\[scheduler\]" "$config_file"; then
    echo "[scheduler] section found in $config_file"
    # Add the lock_path below the [scheduler] section
    sudo sed -i "/^\[scheduler\]/a\$scheduler" "$config_file"
    echo "lock_path added to [scheduler] section"
else
    echo "[scheduler] section not found in $config_file"
    echo "Adding the [scheduler] section and new connection string"
    # If the [scheduler] section doesn't exist, append it to the config file
    echo -e "\n\[scheduler\]\n\$scheduler" | sudo tee -a "$config_file"
fi
else
    echo "$config_file not found."
    exiting....."
    exit 1
fi

```

```

val=$(egrep -c '(vmx|svm)' /proc/cpuinfo)
if [ $val -eq 0 ]; then
    config_file1="/etc/nova/nova-compute.conf"
    libvirt="virt_type = qemu"
    if [ -f "$config_file1" ]; then
        echo "updating the $config_file1"
        sudo cp $config_file1 $config_file2.bak
        # edit libvirt section
        if grep -q "^\[libvirt\]" "$config_file1"; then
            echo "[libvirt] section found in $config_file1"
            # Add the lock_path below the [libvirt] section
            sudo sed -i "/^\[libvirt\]/a$libvirt" "$config_file1"
            echo "lock_path added to [libvirt] section"
        else
            echo "[libvirt] section not found in $config_file1"
            echo "Adding the [libvirt] section and new connection string"
            # If the [libvirt] section doesn't exist, append it to the config file
            echo -e "\n\[libvirt\]\n$libvirt" | sudo tee -a "$config_file1"
        fi
    else
        echo "$config_file1 not found. exiting....."
        exit 1
    fi
else
    echo "using kvm hypervisor"
fi
sudo chown nova:nova -R /etc/nova/
sudo chown nova:nova -R /etc/nova/nova.conf
sudo chmod -R 755 /etc/nova/nova.conf
sudo chown nova:nova -R /var/lib/nova/instances/
sudo chmod -R 755 /var/lib/nova/instances/
# admin authentication for cli
. /home/openstack/admin-openrc
# Populate the nova-api database.
su -s /bin/sh -c "nova-manage api_db sync" nova
# Register the cell0 database.
su -s /bin/sh -c "nova-manage cell_v2 map_cell0" nova
# Create the cell1 cell.
su -s /bin/sh -c "nova-manage cell_v2 create_cell --name=cell1 --verbose" nova
# Populate the nova database.
su -s /bin/sh -c "nova-manage db sync" nova
# Verify nova cell0 and cell1 are registered correctly.
su -s /bin/sh -c "nova-manage cell_v2 list_cells" nova
# on controller node
#Source the admin credentials to enable admin-only CLI commands, then confirm there are compute hosts in
the database:
openstack compute service list --service nova-compute
#Discover compute hosts:
su -s /bin/sh -c "nova-manage cell_v2 discover_hosts --verbose" nova
# Restart the Compute services.
sudo service nova-api restart
sudo service nova-scheduler restart
sudo service nova-conductor restart
sudo service nova-novncproxy restart
sudo service nova-compute restart

```

h) Neutron (Network) Installation

OpenStack Networking (neutron) manages all networking facets for the Virtual Networking Infrastructure (VNI) and the access layer aspects of the Physical Networking Infrastructure (PNI) in your OpenStack environment. OpenStack Networking enables projects to create advanced virtual network topologies which may include services such as a firewall, and a virtual private network (VPN).

```
#!/bin/bash

#---- Neutron Installation-----

DB_ROOT_PASS="123"

SQL_COMMANDS=$(cat <<EOF
CREATE DATABASE neutron;

GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost' IDENTIFIED BY 'ne123';
GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@ '%' IDENTIFIED BY 'ne123';
EOF
)

mysql -uroot -p"$DB_ROOT_PASS" -e "$SQL_COMMANDS"

if [ $? -eq 0 ]; then
    echo "Databases created and privileges granted successfully."
else
    echo "There was an error setting up the databases."
fi
echo "Database setup completed"

. /home/openstack/admin-openrc

# Create the neutron user.
openstack user create --domain default --password "ne123" neutron

# Add the admin role to the neutron user.
openstack role add --project service --user neutron admin

# Create the neutron service entity.
openstack service create --name neutron --description "OpenStack Networking" network

# Create the Networking service API endpoints.
openstack endpoint create --region RegionOne network public http://controller:9696
openstack endpoint create --region RegionOne network admin http://controller:9696
openstack endpoint create --region RegionOne network internal http://controller:9696

# install the neutron packages.
sudo apt install neutron-server neutron-plugin-ml2 neutron-openvswitch-agent neutron-l3-agent neutron-dhcp-agent neutron-metadata-agent -y

# Edit the /etc/neutron/neutron.conf file and complete the following actions. In the [database] section,
configure database access:
config_file="/etc/neutron/neutron.conf"
db_con="connection = mysql+pymysql://neutron:ne123@controller/neutron"
def_ser="core_plugin = ml2"
```

```

service_plugins = router
auth_strategy = keystone
notify_nova_on_port_status_changes = true
notify_nova_on_port_data_changes = true"
def_trans="transport_url = rabbit://openstack:R123@controller"
keystone_authtoken_config="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 = neutron
password = ne123"
nova_conf="auth_url = http://controller:5000
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
project_name = service
username = nova
password = no123"
oslo_conf="lock_path = /var/lib/neutron/tmp"

config_file1="/etc/neutron/plugins/ml2/ml2_conf.ini"
ml2_conf="type_drivers = flat,vlan,vxlan
tenant_network_types = vxlan
mechanism_drivers = openvswitch,l2population
extension_drivers = port_security"
ml2_type_flat="flat_networks = provider"
ml2_type_vxlan="vni_ranges = 1:1000"

config_file2="/etc/neutron/plugins/ml2/openvswitch_agent.ini"
ovs="bridge_mappings = provider:br-ex
local_ip = 10.0.2.15"
agent="tunnel_types = vxlan
l2_population = true"
securitygroup="enable_security_group = true
firewall_driver = openvswitch"

config_file3="/etc/neutron/l3_agent.ini"
def1="interface_driver = openvswitch"

config_file4="/etc/neutron/dhcp_agent.ini"
def="interface_driver = openvswitch
dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq
enable_isolated_metadata = true"

config_file5="/etc/neutron/metadata_agent.ini"
def2="nova_metadata_host = controller
metadata_proxy_shared_secret = fe6702d1e7648ffce39f"

if [ -f "$config_file" ]; then
    echo "updating the $config_file"
    sudo cp $config_file $config_file.bak

```

```

#In the [database] and [database] sections, configure database access.
if grep -q "^\[database\]" "$config_file"; then
    echo "[database] section found in $config_file"
    # Check if there is already a connection string in the [api_database] section
    if grep -A1 "^\[database\]" "$config_file" | grep -q "^connection ="; then
        echo "Existing connection string found, commenting it out"
        # Comment out the existing connection string
        sudo sed -i "s/^.*connection = .*/# &/" "$config_file"
    else
        echo "No existing connection string found in the [database] section"
    fi
    # Add the new connection string below the [api_database] section
    sudo sed -i "/^\[database\]/a$db_con" "$config_file"
    echo "New connection string added to [database] section"
else
    echo "[database] section not found in $config_file"
    echo "Adding the [database] section and new connection string"
    # If the [database] section doesn't exist, append it to the config file
    echo -e "\n[database]\n$db_con" | sudo tee -a "$config_file"
fi
# Editing the [DEFAULT] section
if grep -q "^\[DEFAULT\]" "$config_file"; then
    echo "[DEFAULT] section found in $config_file"
    # Add the keystone_authtoken below the [DEFAULT] section
    while IFS= read -r line; do
        key=$(echo "$line" | cut -d=' ' -f1 | xargs)
        value=$(echo "$line" | cut -d=' ' -f2- | xargs)
        # Check if the key exists in the section, if not, add it
        if grep -A10 "^\[DEFAULT\]" "$config_file" | grep -q "^$key ="; then
            echo "Updating $key in DEFAULT section"
            sudo sed -i "/^\[DEFAULT\]/,/^\[/ s/^$key =.*|$key = $value|" "$config_file"
        else
            echo "Adding $key to DEFAULT section"
            sudo sed -i "/^\[DEFAULT\]/a$key = $value" "$config_file"
        fi
    done <<< "$def_ser"
    echo "default services added to [DEFAULT] section"
else
    echo "[DEFAULT] section not found in $config_file"
    echo "Adding the [DEFAULT] section and new connection string"
    # If the [DEFAULT] section doesn't exist, append it to the config file
    echo "[DEFAULT]" >> "$config_file"

    # Add the DEFAULT below the [DEFAULT] section
    while IFS= read -r line; do
        key=$(echo "$line" | cut -d=' ' -f1 | xargs)
        value=$(echo "$line" | cut -d=' ' -f2- | xargs)
        # Check if the key exists in the section, if not, add it
        if grep -A10 "^\[DEFAULT\]" "$config_file" | grep -q "^$key ="; then
            echo "Updating $key in DEFAULT section"
            sudo sed -i "/^\[DEFAULT\]/,/^\[/ s|^$key =.*|$key = $value|" "$config_file"
        else
            echo "Adding $key to DEFAULT section"
            sudo sed -i "/^\[DEFAULT\]/a$key = $value" "$config_file"
        fi
    done <<< "$def_ser"
fi

```

```

# Check if the [DEFAULT] section exists
if grep -q "^\[DEFAULT\]" "$config_file"; then
    echo "[DEFAULT] section found in $config_file"

    # Check if there is already a transport_url string in the [Default] section
    if grep -A1 "^\[DEFAULT\]" "$config_file" | grep -q "^transport_url ="; then
        echo "Existing transport_url string found, commenting it out"
        # Comment out the existing transport_url string
        sudo sed -i "s/^\.*transport_url = .*/# &/" "$config_file"
    else
        echo "No existing connection string found in the [default] section"
    fi
    # Add the new connection string below the [database] section
    sudo sed -i "^\[DEFAULT\]/a$def_trans" "$config_file"
    echo "New connection string added to [DEFAULT] section"
else
    echo "[DEFAULT] section not found in $config_file"
    echo "Adding the [DEFAULT] section and new connection string"
    # If the [DEFAULT] section doesn't exist, append it to the config file
    echo -e "\n\[DEFAULT\]\n\n$def_trans" | sudo tee -a "$config_file"
fi
# Editing the [keystone_authtoken] section
if grep -q "^\[keystone_authtoken\]" "$config_file"; then
    echo "[keystone_authtoken] section found in $config_file"
    # Add the keystone_authtoken below the [keystone_authtoken] section
    while IFS= read -r line; do
        key=$(echo "$line" | cut -d=' ' -f1 | xargs)
        value=$(echo "$line" | cut -d=' ' -f2- | xargs)
        # Check if the key exists in the section, if not, add it
        if grep -A10 "^\[keystone_authtoken\]" "$config_file" | grep -q "^$key ="; then
            echo "Updating $key in keystone_authtoken section"
            sudo sed -i "^\[keystone_authtoken\]/,/^[/ s/^$key =.*|$key = $value|" "$config_file"
        else
            echo "Adding $key to keystone_authtoken section"
            sudo sed -i "^\[keystone_authtoken\]/a$key = $value" "$config_file"
        fi
    done <<< "$keystone_authtoken_config"
    echo "keystone_authtoken added to [keystone_authtoken] section"
else
    echo "[keystone_authtoken] section not found in $config_file"
    echo "Adding the [keystone_authtoken] section and new connection string"
    # If the [keystone_authtoken] section doesn't exist, append it to the config file
    echo "[keystone_authtoken]" >> "$config_file"
    # Add the keystone_authtoken below the [keystone_authtoken] section
    while IFS= read -r line; do
        key=$(echo "$line" | cut -d=' ' -f1 | xargs)
        value=$(echo "$line" | cut -d=' ' -f2- | xargs)
        # Check if the key exists in the section, if not, add it
        if grep -A10 "^\[keystone_authtoken\]" "$config_file" | grep -q "^$key ="; then
            echo "Updating $key in keystone_authtoken section"
            sudo sed -i "^\[keystone_authtoken\]/,/^[/ s/^$key =.*|$key = $value|" "$config_file"
        else
            echo "Adding $key to keystone_authtoken section"
            sudo sed -i "^\[keystone_authtoken\]/a$key = $value" "$config_file"
        fi
    done <<< "$keystone_authtoken_config"
fi

```



```

# Editing the [nova] section
if grep -q "^\[nova\]" "$config_file"; then
    echo "[nova] section found in $config_file"

    # Add the nova config below in the [nova] section
    while IFS= read -r line; do
        key=$(echo "$line" | cut -d=' ' -f1 | xargs)
        value=$(echo "$line" | cut -d=' ' -f2- | xargs)

        # Check if the key exists in the section, if not, add it
        if grep -A10 "^\[nova\]" "$config_file" | grep -q "^$key ="; then
            echo "Updating $key in nova section"
            sudo sed -i "s/^\[nova\]/,/\[ / s/^$key =.*|$key = $value|" "$config_file"
        else
            echo "Adding $key to nova section"
            sudo sed -i "s/^\[nova\]/a$key = $value" "$config_file"
        fi
    done <<< "$nova_conf"
    echo "nova configuration added to [nova] section"
else
    echo "[nova] section not found in $config_file"
    echo "Adding the [nova] section and new connection string"
    # If the [nova] section doesn't exist, append it to the config file
    echo "[nova]" >> "$config_file"
    # Add the nova configuration below the [nova] section
    while IFS= read -r line; do
        key=$(echo "$line" | cut -d=' ' -f1 | xargs)
        value=$(echo "$line" | cut -d=' ' -f2- | xargs)
        # Check if the key exists in the section, if not, add it
        if grep -A10 "^\[nova\]" "$config_file" | grep -q "^$key ="; then
            echo "Updating $key in nova section"
            sudo sed -i "s/^\[nova\]/,/\[ / s/^$key =.*|$key = $value|" "$config_file"
        else
            echo "Adding $key to nova section"
            sudo sed -i "s/^\[nova\]/a$key = $value" "$config_file"
        fi
    done <<< "$nova_conf"
fi
# edit oslo_concurrency section
if grep -q "^\[oslo_concurrency\]" "$config_file"; then
    echo "[oslo_concurrency] section found in $config_file"
    # Add the api_server below the [glance] section
    sudo sed -i "s/^\[oslo_concurrency\]/a$oslo_conf" "$config_file"
    echo "oslo_concurrency added to [oslo_concurrency] section"
else
    echo "[oslo_concurrency] section not found in $config_file"
    echo "Adding the [oslo_concurrency] section and new connection string"

    # If the [oslo_concurrency] section doesn't exist, append it to the config file
    echo -e "\n[oslo_concurrency]\n$oslo_conf" | sudo tee -a "$config_file"
fi
else
    echo "$config_file not found."
    exiting....."
    exit 1
fi

```

```

# configure the $config_file1. check that file was found or not
if [ -f "$config_file1" ]; then
    echo "updating the $config_file1"
    sudo cp $config_file1 $config_file1.bak
    # Editing the [ml2] section
    if grep -q "^\[ml2\]" "$config_file1"; then
        echo "[ml2] section found in $config_file1"
        # Add the nova config below in the [nova] section
        while IFS= read -r line; do
            key=$(echo "$line" | cut -d=' ' -f1 | xargs)
            value=$(echo "$line" | cut -d=' ' -f2- | xargs)
            # Check if the key exists in the section, if not, add it
            if grep -A10 "^\[ml2\]" "$config_file1" | grep -q "^$key ="; then
                echo "Updating $key in ml2 section"
                sudo sed -i "/^\[ml2\]/,/^[/ s/^$key =.*|$key = $value|" "$config_file1"
            else
                echo "Adding $key to ml2 section"
                sudo sed -i "/^\[ml2\]/a$key = $value" "$config_file1"
            fi
        done <<< "$ml2_conf"
        echo "nova configuration added to [ml2] section"
    else
        echo "[ml2] section not found in $config_file1"
        echo "Adding the [ml2] section and new connection string"
        # If the [ml2] section doesn't exist, append it to the config file
        echo "[ml2]" >> "$config_file1"
        # Add the ml2 below the [ml2] section
        while IFS= read -r line; do
            key=$(echo "$line" | cut -d=' ' -f1 | xargs)
            value=$(echo "$line" | cut -d=' ' -f2- | xargs)
            # Check if the key exists in the section, if not, add it
            if grep -A10 "^\[ml2\]" "$config_file1" | grep -q "^$key ="; then
                echo "Updating $key in ml2 section"
                sudo sed -i "/^\[ml2\]/,/^[/ s/^$key =.*|$key = $value|" "$config_file1"
            else
                echo "Adding $key to ml2 section"
                sudo sed -i "/^\[ml2\]/a$key = $value" "$config_file1"
            fi
        done <<< "$ml2_conf"
    fi
    # edit ml2_type_flat section
    if grep -q "^\[ml2_type_flat\]" "$config_file1"; then
        echo "[ml2_type_flat] section found in $config_file1"
        # Add the ml2_type_flat below the [ml2_type_flat] section
        sudo sed -i "/^\[ml2_type_flat\]/a$ml2_type_flat" "$config_file1"
        echo "ml2_type_flat added to [ml2_type_flat] section"
    else
        echo "[ml2_type_flat] section not found in $config_file1"
        echo "Adding the [ml2_type_flat] section and new connection string"

        # If the [ml2_type_flat] section doesn't exist, append it to the config file
        echo -e "\n[ml2_type_flat]\n$ml2_type_flat" | sudo tee -a "$config_file1"
    fi
    # edit ml2_type_vxlan section
    if grep -q "^\[ml2_type_vxlan\]" "$config_file1"; then
        echo "[ml2_type_vxlan] section found in $config_file1"
    fi

```

```

        # Add the ml2_type_vxlan below the [ml2_type_vxlan] section
        sudo sed -i "/^\[ml2_type_vxlan\]/a$ml2_type_vxlan" "$config_file1"
        echo "ml2_type_vxlan added to [ml2_type_vxlan] section"
    else
        echo "[ml2_type_vxlan] section not found in $config_file1"
        echo "Adding the [ml2_type_vxlan] section and new connection string"
        # If the [ml2_type_vxlan] section doesn't exist, append it to the config file
        echo -e "\n[ml2_type_vxlan]\n$ml2_type_vxlan" | sudo tee -a "$config_file1"
    fi
else
    echo "$config_file1 not found."
    exiting....."
    exit 1
fi
# configure the $config_file2. check that file was found or not
if [ -f "$config_file2" ]; then
    echo "updating the $config_file2"
    sudo cp $config_file2 $config_file2.bak
    # Editing the [ovs] section
    if grep -q "^\[ovs\]" "$config_file2"; then
        echo "[ovs] section found in $config_file2"

        # Add the nova config below in the [ovs] section
        while IFS= read -r line; do
            key=$(echo "$line" | cut -d=' ' -f1 | xargs)
            value=$(echo "$line" | cut -d=' ' -f2- | xargs)

            # Check if the key exists in the section, if not, add it
            if grep -A10 "^\[ovs\]" "$config_file2" | grep -q "^$key ="; then
                echo "Updating $key in ovs section"
                sudo sed -i "/^\[ovs\]/,/^\[/ s/^$key =.*|$key = $value|" "$config_file2"
            else
                echo "Adding $key to ovs section"
                sudo sed -i "/^\[ovs\]/a$key = $value" "$config_file2"
            fi
        done <<< "$ovs"
        echo "ovs configuration added to [ovs] section"
    else
        echo "[ovs] section not found in $config_file2"
        echo "Adding the [ovs] section and new connection string"

        # If the [ovs] section doesn't exist, append it to the config file
        echo "[ovs]" >> "$config_file2"
        # Add the ovs configuration below the [ovs] section
        while IFS= read -r line; do
            key=$(echo "$line" | cut -d=' ' -f1 | xargs)
            value=$(echo "$line" | cut -d=' ' -f2- | xargs)
            # Check if the key exists in the section, if not, add it
            if grep -A10 "^\[ovs\]" "$config_file2" | grep -q "^$key ="; then
                echo "Updating $key in ovs section"
                sudo sed -i "/^\[ovs\]/,/^\[/ s/^$key =.*|$key = $value|" "$config_file2"
            else
                echo "Adding $key to ovs section"
                sudo sed -i "/^\[ovs\]/a$key = $value" "$config_file2"
            fi
        done <<< "$ovs"
    fi
fi

```

```

# Editing the [agent] section
if grep -q "\[agent\]" "$config_file2"; then
    echo "[agent] section found in $config_file2"
    # Add the agent config below in the [agent] section
    while IFS= read -r line; do
        key=$(echo "$line" | cut -d=' ' -f1 | xargs)
        value=$(echo "$line" | cut -d=' ' -f2- | xargs)
        # Check if the key exists in the section, if not, add it
        if grep -A10 "\[agent\]" "$config_file2" | grep -q "^$key ="; then
            echo "Updating $key in agent section"
            sudo sed -i "\[agent\]/,/\[/ s|^$key =.*|$key = $value|" "$config_file2"
        else
            echo "Adding $key to agent section"
            sudo sed -i "\[agent\]/a$key = $value" "$config_file2"
        fi
    done <<< "$agent"
    echo "agent configuration added to [agent] section"
else
    echo "[agent] section not found in $config_file2"
    echo "Adding the [agent] section and new connection string"

# If the [agent] section doesn't exist, append it to the config file
echo "[agent]" >> "$config_file2"
# Add the agent configuration below the [agent] section
while IFS= read -r line; do
    key=$(echo "$line" | cut -d=' ' -f1 | xargs)
    value=$(echo "$line" | cut -d=' ' -f2- | xargs)
    # Check if the key exists in the section, if not, add it
    if grep -A10 "\[agent\]" "$config_file2" | grep -q "^$key ="; then
        echo "Updating $key in agent section"
        sudo sed -i "\[agent\]/,/\[/ s|^$key =.*|$key = $value|" "$config_file2"
    else
        echo "Adding $key to agent section"
        sudo sed -i "\[agent\]/a$key = $value" "$config_file2"
    fi
done <<< "$agent "
fi

# Editing the [securitygroup] section
if grep -q "\[securitygroup\]" "$config_file2"; then
    echo "[securitygroup] section found in $config_file2"

    # Add the agent config below in the [agent] section
    while IFS= read -r line; do
        key=$(echo "$line" | cut -d=' ' -f1 | xargs)
        value=$(echo "$line" | cut -d=' ' -f2- | xargs)

        # Check if the key exists in the section, if not, add it
        if grep -A10 "\[securitygroup\]" "$config_file2" | grep -q "^$key ="; then
            echo "Updating $key in securitygroup section"
            sudo sed -i "\[securitygroup\]/,/\[/ s|^$key =.*|$key = $value|" "$config_file2"
        else
            echo "Adding $key to securitygroup section"
            sudo sed -i "\[securitygroup\]/a$key = $value" "$config_file2"
        fi
    done <<< "$securitygroup"
    echo "securitygroup configuration added to [securitygroup] section"

```

```

else
    echo "[securitygroup] section not found in $config_file2"
    echo "Adding the [securitygroup] section and new connection string"

# If the [securitygroup] section doesn't exist, append it to the config file
echo "[securitygroup]" >> "$config_file2"

# Add the securitygroup configuration below the [securitygroup] section
while IFS= read -r line; do
    key=$(echo "$line" | cut -d'=' -f1 | xargs)
    value=$(echo "$line" | cut -d'=' -f2- | xargs)
    # Check if the key exists in the section, if not, add it
    if grep -A10 "\[securitygroup\]" "$config_file2" | grep -q "^$key ="; then
        echo "Updating $key in securitygroup section"
        sudo sed -i "/\[securitygroup\]/,/^\[ s\|^$key =.*$key = $value|" "$config_file2"
    else
        echo "Adding $key to securitygroup section"
        sudo sed -i "/\[securitygroup\]/a$key = $value" "$config_file2"
    fi
done <<< "$securitygroup"
fi

sudo sysctl net.bridge.bridge-nf-call-iptables = 1
sudo sysctl net.bridge.bridge-nf-call-ip6tables = 1
sudo ovs-vsctl add-br br-ex
sudo ovs-vsctl add-port br-ex PhysicalInterface # provide the physical interface
sudo ifconfig up br-ex
sudo ip addr add IPAddress/Subnet dev br-ex # Change the Ip Address with provider ip
sudo ip addr flush enp0s8
sudo ip link set br-ex
sudo ip route add default via Gateway-address dev br-ex proto dhcp metric 100 # change the gateway
address
else
    echo "$config_file2 not found."
    exiting....."
    exit 1
fi
# configure the $config_file3. check that file was found or not
if [ -f "$config_file3" ]; then
    echo "updating the $config_file3"
    sudo cp $config_file3 $config_file3.bak

    # edit DEFAULT section
    if grep -q "\[DEFAULT\]" "$config_file3"; then
        echo "[DEFAULT] section found in $config_file3"

        # Add the DEFAULT below the [DEFAULT] section
        sudo sed -i "/\[DEFAULT\]/a$def1" "$config_file3"
        echo "DEFAULTs added to [DEFAULT] section"
    else
        echo "[DEFAULT] section not found in $config_file3"
        echo "Adding the [DEFAULT] section and new connection string"

        # If the [DEFAULT] section doesn't exist, append it to the config file
        echo -e "\n[DEFAULT]\n$def1" | sudo tee -a "$config_file3"
    fi

```

```

else
    echo "$config_file3 not found.
    exiting....."
    exit 1
fi

# configure the $config_file4. check that file was found or not
if [ -f "$config_file4" ]; then
    echo "updating the $config_file4"
    sudo cp $config_file4 $config_file4.bak

    # Editing the [DEFAULT] section
    if grep -q "^\[DEFAULT\]" "$config_file4"; then
        echo "[DEFAULT] section found in $config_file4"

        # Add the agent config below in the [DEFAULT] section
        while IFS= read -r line; do
            key=$(echo "$line" | cut -d=' ' -f1 | xargs)
            value=$(echo "$line" | cut -d=' ' -f2- | xargs)

            # Check if the key exists in the section, if not, add it
            if grep -A10 "^\[DEFAULT\]" "$config_file4" | grep -q "^\$key ="; then
                echo "Updating $key in DEFAULT section"
                sudo sed -i "/^\[DEFAULT\]/,/^\[/ s/^\$key =.*|\$key = \$value|" "$config_file4"
            else
                echo "Adding $key to DEFAULT section"
                sudo sed -i "/^\[DEFAULT\]/a\$key = \$value" "$config_file4"
            fi
        done <<< "$def"
        echo "DEFAULT configuration added to [DEFAULT] section"

    else
        echo "[DEFAULT] section not found in $config_file4"
        echo "Adding the [DEFAULT] section and new connection string"

        # If the [DEFAULT] section doesn't exist, append it to the config file
        echo "[DEFAULT]" >> "$config_file4"

        # Add the securitygroup configuration below the [securitygroup] section
        while IFS= read -r line; do
            key=$(echo "$line" | cut -d=' ' -f1 | xargs)
            value=$(echo "$line" | cut -d=' ' -f2- | xargs)
            # Check if the key exists in the section, if not, add it
            if grep -A10 "^\[DEFAULT\]" "$config_file4" | grep -q "^\$key ="; then
                echo "Updating $key in DEFAULT section"
                sudo sed -i "/^\[DEFAULT\]/,/^\[/ s/^\$key =.*|\$key = \$value|" "$config_file4"
            else
                echo "Adding $key to DEFAULT section"
                sudo sed -i "/^\[DEFAULT\]/a\$key = \$value" "$config_file4"
            fi
        done <<< "$def "
    fi
fi

else
    echo "$config_file4 not found.
    exiting....."
    exit 1
fi

```

```

# configure the $config_file5. check that file was found or not
if [ -f "$config_file5" ]; then
    echo "updating the $config_file5"
    sudo cp $config_file5 $config_file5.bak
    # Editing the [DEFAULT] section
    if grep -q "^\[DEFAULT\]" "$config_file5"; then
        echo "[DEFAULT] section found in $config_file5"
        # Add the agent config below in the [DEFAULT] section
        while IFS= read -r line; do
            key=$(echo "$line" | cut -d=' ' -f1 | xargs)
            value=$(echo "$line" | cut -d=' ' -f2- | xargs)
            # Check if the key exists in the section, if not, add it
            if grep -A10 "^\[DEFAULT\]" "$config_file5" | grep -q "^$key ="; then
                echo "Updating $key in DEFAULT section"
                sudo sed -i "/^\[DEFAULT\]/,/^\[/ s/^$key =.*|$key = $value|" "$config_file5"
            else
                echo "Adding $key to DEFAULT section"
                sudo sed -i "/^\[DEFAULT\]/a$key = $value" "$config_file5"
            fi
        done <<< "$def2"
        echo "DEFAULT configuration added to [DEFAULT] section"
    else
        echo "[DEFAULT] section not found in $config_file5"
        echo "Adding the [DEFAULT] section and new connection string"
        # If the [DEFAULT] section doesn't exist, append it to the config file
        echo "[DEFAULT]" >> "$config_file5"
        # Add the securitygroup configuration below the [securitygroup] section
        while IFS= read -r line; do
            key=$(echo "$line" | cut -d=' ' -f1 | xargs)
            value=$(echo "$line" | cut -d=' ' -f2- | xargs)
            # Check if the key exists in the section, if not, add it
            if grep -A10 "^\[DEFAULT\]" "$config_file5" | grep -q "^$key ="; then
                echo "Updating $key in DEFAULT section"
                sudo sed -i "/^\[DEFAULT\]/,/^\[/ s/^$key =.*|$key = $value|" "$config_file5"
            else
                echo "Adding $key to DEFAULT section"
                sudo sed -i "/^\[DEFAULT\]/a$key = $value" "$config_file5"
            fi
        done <<< "$def2"
    fi
else
    echo "$config_file5 not found."
    exiting....."
    exit 1
fi

# Populate the 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
# For both networking options:
service neutron-server restart
service neutron-openvswitch-agent restart
service neutron-dhcp-agent restart
service neutron-metadata-agent restart

#For networking option 2, also restart the layer-3 service:
service neutron-l3-agent restart

```

```
root@c3ihub: /home/k8# route -n
Kernel IP routing table
Destination      Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0          192.168.42.2   0.0.0.0         UG    100    0      0 br-ex1
0.0.0.0          192.168.42.2   0.0.0.0         UG    100    0      0 br-storage
0.0.0.0          172.29.235.254 0.0.0.0         UG    100    0      0 br-ex
0.0.0.0          192.168.42.2   0.0.0.0         UG    100    0      0 eno12399
172.20.160.23    192.168.42.2   255.255.255.255 UGH    100    0      0 eno12399
172.29.232.0     0.0.0.0        255.255.252.0   U    100    0      0 br-ex
172.31.1.1       192.168.42.2   255.255.255.255 UGH    100    0      0 eno12399
172.31.1.130     192.168.42.2   255.255.255.255 UGH    100    0      0 eno12399
192.168.42.0     0.0.0.0        255.255.255.0   U    100    0      0 br-ex1
192.168.42.0     0.0.0.0        255.255.255.0   U    100    0      0 br-storage
192.168.42.0     0.0.0.0        255.255.255.0   U    100    0      0 eno12399
192.168.42.2     0.0.0.0        255.255.255.255 UH    100    0      0 eno12399
192.168.122.0    0.0.0.0        255.255.255.0   U    0      0      0 virbr0
root@c3ihub: /home/k8#
```

i) Cinder (Block/Volume) installation

The Block Storage service (cinder) provides block storage devices to guest instances. The method in which the storage is provisioned and consumed is determined by the Block Storage driver, or drivers in the case of a multi-backend configuration.

This script contains the controller node as controller node as well as storage node. It will make the hard disk of the controller node for creating the volume s on openstack.

```
#!/bin/bash

#---controller--
DB_ROOT_PASS="123"

SQL_COMMANDS=$(cat <<EOF
CREATE DATABASE cinder;
GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'localhost' IDENTIFIED BY 'ci123';
GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'%' IDENTIFIED BY 'ci123';
EOF
)

mysql -uroot -p"$DB_ROOT_PASS" -e "$SQL_COMMANDS"

if [ $? -eq 0 ]; then
    echo "Databases created and privileges granted successfully."
else
    echo "There was an error setting up the databases."
fi
echo "Database setup completed"

. /home/openstack/admin-openrc
#Create a cinder user:
openstack user create --domain default --password "ci123" cinder

# Add the admin role to the cinder user:
openstack role add --project service --user cinder admin
# Create the cinderv3 service entity.
openstack service create --name cinderv3 --description "OpenStack Block Storage" volumev3
```



```

# Create the Block Storage service API endpoints:
openstack endpoint create --region RegionOne volumev3 public http://controller:8776/v3/%(project_id)s
openstack endpoint create --region RegionOne volumev3 internal http://controller:8776/v3/%(project_id)s
openstack endpoint create --region RegionOne volumev3 admin http://controller:8776/v3/%(project_id)s
# install the package
sudo apt install cinder-api cinder-scheduler -y
# Edit the /etc/cinder/cinder.conf file and complete the following actions:
config_file="/etc/cinder/cinder.conf"
db_con="connection = mysql+pymysql://cinder:ci123@controller/cinder"
trans_url="transport_url = rabbit://openstack:R123@controller"
def_ser="auth_strategy = keystone"
keystone_auth_token_config="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 = cinder
password = ci123
service_token_roles_required = True
service_token_roles = admin"
oslo_con="lock_path = /var/lib/cinder/tmp"
if [ -f "$config_file" ]; then
    echo "updating the $config_file"
    sudo cp $config_file $config_file.bak
    sudo chown cinder:cinder /etc/cinder/cinder.conf
    #In the [database] and [database] sections, configure database access.
    if grep -q "^\[database\]" "$config_file"; then
        echo "[database] section found in $config_file"
        # Check if there is already a connection string in the [api_database] section
        if grep -A1 "^\[database\]" "$config_file" | grep -q "^connection ="; then
            echo "Existing connection string found, commenting it out"
            # Comment out the existing connection string
            sudo sed -i "s/^.*connection = .*/# &/" "$config_file"
        else
            echo "No existing connection string found in the [database] section"
        fi
        # Add the new connection string below the [api_database] section
        sudo sed -i "s/^\[database\]/a$db_con" "$config_file"
        echo "New connection string added to [database] section"
    else
        echo "[database] section not found in $config_file"
        echo "Adding the [database] section and new connection string"
        # If the [database] section doesn't exist, append it to the config file
        echo -e "\n[database]\n$db_con" | sudo tee -a "$config_file"
    fi

    #In the [DEFAULT] section, configure RabbitMQ message queue access.
    if grep -q "^\[DEFAULT\]" "$config_file"; then
        echo "[DEFAULT] section found in $config_file"

        # Check if there is already a transport_url string in the [Default] section
        if grep -A1 "^\[DEFAULT\]" "$config_file" | grep -q "^transport_url ="; then
            echo "Existing transport_url string found, commenting it out"
            # Comment out the existing transport_url string
            sudo sed -i "s/^.*transport_url = .*/# &/" "$config_file"
        fi
    fi
fi

```

```

else
    echo "No existing connection string found in the [default] section"
fi

# Add the new connection string below the [database] section
sudo sed -i "/^\[DEFAULT\]/a$trans_url" "$config_file"
echo "New connection string added to [DEFAULT] section"

else
    echo "[DEFAULT] section not found in $config_file"
    echo "Adding the [DEFAULT] section and new connection string"

# If the [DEFAULT] section doesn't exist, append it to the config file
echo -e "\n[DEFAULT]\n\n$trans_url" | sudo tee -a "$config_file"
fi

# edit the [DEFAULT] section for keystone
if grep -q "^\[DEFAULT\]" "$config_file"; then
    echo "[DEFAULT] section found in $config_file"

    # Add the DEFAULT below the [DEFAULT] section
    sudo sed -i "/^\[DEFAULT\]/a$def_ser" "$config_file"
    echo "auth_strategy added to [DEFAULT] section"

else
    echo "[DEFAULT] section not found in $config_file"
    echo "Adding the [DEFAULT] section and keystone auth"

# If the [DEFAULT] section doesn't exist, append it to the config file
echo -e "\n[DEFAULT]\n\n$def_ser" | sudo tee -a "$config_file"
fi

# Editing the [keystone_authtoken] section
if grep -q "^\[keystone_authtoken\]" "$config_file"; then
    echo "[keystone_authtoken] section found in $config_file"

    # Add the keystone_authtoken below the [keystone_authtoken] section
    while IFS= read -r line; do
        key=$(echo "$line" | cut -d=' ' -f1 | xargs)
        value=$(echo "$line" | cut -d=' ' -f2- | xargs)

        # Check if the key exists in the section, if not, add it
        if grep -A10 "^\[keystone_authtoken\]" "$config_file" | grep -q "^$key ="; then
            echo "Updating $key in keystone_authtoken section"
            sudo sed -i "/^\[keystone_authtoken\]/,/^\[ s|^$key =.*|$key = $value|" "$config_file"
        else
            echo "Adding $key to keystone_authtoken section"
            sudo sed -i "/^\[keystone_authtoken\]/a$key = $value" "$config_file"
        fi
    done <<< "$keystone_authtoken_config"
    echo "keystone_authtoken added to [keystone_authtoken] section"
else
    echo "[keystone_authtoken] section not found in $config_file"
    echo "Adding the [keystone_authtoken] section and new connection string"

# If the [keystone_authtoken] section doesn't exist, append it to the config file
echo "[keystone_authtoken]" >> "$config_file"

```

```

# Add the keystone_auth token below the [keystone_auth token] section
while IFS= read -r line; do
    key=$(echo "$line" | cut -d=' ' -f1 | xargs)
    value=$(echo "$line" | cut -d=' ' -f2- | xargs)
    # Check if the key exists in the section, if not, add it
    if grep -A10 "\[keystone_auth token\]" "$config_file" | grep -q "^$key ="; then
        echo "Updating $key in keystone_auth token section"
        sudo sed -i "\[keystone_auth token\]/,/\[/ s/^$key =.*|$key = $value|" "$config_file"
    else
        echo "Adding $key to keystone_auth token section"
        sudo sed -i "\[keystone_auth token\]/a$key = $value" "$config_file"
    fi
done <<< "$keystone_auth token_config"
fi
# edit oslo_concurrency section
if grep -q "\[oslo_concurrency\]" "$config_file"; then
    echo "[oslo_concurrency] section found in $config_file"

    # Add the lock_path below the [oslo_concurrency] section
    sudo sed -i "\[oslo_concurrency\]/a$oslo_con" "$config_file"
    echo "lock_path added to [oslo_concurrency] section"

else
    echo "[oslo_concurrency] section not found in $config_file"
    echo "Adding the [oslo_concurrency] section and new connection string"

    # If the [oslo_concurrency] section doesn't exist, append it to the config file
    echo -e "\n[oslo_concurrency]\n$oslo_con" | sudo tee -a "$config_file"
fi

else
    echo "$config_file not found."
    exiting....."
    exit 1
fi

# Populate the Block Storage database.
su -s /bin/sh -c "cinder-manage db sync" cinder
config_file1="/etc/nova/nova.conf"
cinder="os_region_name = RegionOne"
conn="/etc/nova/nova-cinder.conf"
if [ -f "$config_file1" ]; then
    echo "updating the $config_file1"
    sudo cp $config_file1 $conn.bak

    # edit cinder section
    if grep -q "\[cinder\]" "$config_file1"; then
        echo "[cinder] section found in $config_file1"
        # Add the cinder below the [cinder] section
        sudo sed -i "\[cinder\]/a$oslo_con" "$config_file1"
        echo "region_name to [cinder] section"
    else
        echo "[cinder] section not found in $config_file1"
        echo "Adding the [cinder] section and new connection string"

        # If the [cinder] section doesn't exist, append it to the config file
        echo -e "\n[cinder]\n$cinder" | sudo tee -a "$config_file1"
    fi

```

```

else
    echo "$config_file1 not found.
    exiting....."
    exit 1
fi

#Restart the Compute API service:
service nova-api restart

#Restart the Block Storage services:
service cinder-scheduler restart
service apache2 restart
echo "services restarted successfully, controller node setup done !!!"

#-----Compute Node Setup-----
# install supporting utility packages.
sudo apt install lvm2 thin-provisioning-tools -y

# Create the LVM physical volume /dev/sdb:
sudo pvcreate /dev/sdb

# Create the LVM volume group cinder-volumes:
sudo vgcreate cinder-volumes /dev/sdb
conf="/etc/lvm/lvm.conf"
sudo cp $conf $conf.bak
sed -i '/devices {/!b;n;c\ filter = [ "a/sdb/", "r/*/" ] $conf
echo "lvm conf done...."

#Install the packages:
sudo apt install cinder-volume tgt -y
lvm="volume_driver = cinder.volume.drivers.lvm.LVMVolumeDriver
volume_group = cinder-volumes
target_protocol = iscsi
target_helper = tgtadm"
def="enabled_backends = lvm
glance_api_servers = http://controller:9292"
if [ -f "$config_file" ]; then
    echo "updating the $config_file"
    sudo cp $config_file $config_file.1.bak
    # Editing the [lvm] section
    if grep -q "^\[lvm\]" "$config_file"; then
        echo "[lvm] section found in $config_file"
        # Add the lvm below the [lvm] section
        while IFS= read -r line; do
            key=$(echo "$line" | cut -d=' ' -f1 | xargs)
            value=$(echo "$line" | cut -d=' ' -f2- | xargs)

            # Check if the key exists in the section, if not, add it
            if grep -A10 "^\[lvm\]" "$config_file" | grep -q "^\$key ="; then
                echo "Updating $key in lvm section"
                sudo sed -i "/^\[lvm\]/,/^\[ / s/^\$key =.*|\$key = \$value|" "$config_file"
            else
                echo "Adding $key to lvm section"
                sudo sed -i "/^\[lvm\]/a\$key = \$value" "$config_file"
            fi
        done <<< "$lvm"
        echo "lvm added to [lvm] section"
    fi

```

```

else
    echo "[lvm] section not found in $config_file"
    echo "Adding the [lvm] section and new connection string"
    # If the [lvm] section doesn't exist, append it to the config file
    echo "[lvm]" >> "$config_file"

    # Add the lvm below the [lvm] section
    while IFS= read -r line; do
        key=$(echo "$line" | cut -d=' ' -f1 | xargs)
        value=$(echo "$line" | cut -d=' ' -f2- | xargs)
        # Check if the key exists in the section, if not, add it
        if grep -A10 "^\[lvm\]" "$config_file" | grep -q "^$key ="; then
            echo "Updating $key in lvm section"
            sudo sed -i "/^\[lvm\]/,/^\[/ s/^$key =.*|$key = $value|" "$config_file"
        else
            echo "Adding $key to lvm section"
            sudo sed -i "/^\[lvm\]/a$key = $value" "$config_file"
        fi
    done <<< "$lvm"
fi

# Editing the [DEFAULT] section
if grep -q "^\[DEFAULT\]" "$config_file"; then
    echo "[DEFAULT] section found in $config_file"
    # Add the lvm below the [DEFAULT] section
    while IFS= read -r line; do
        key=$(echo "$line" | cut -d=' ' -f1 | xargs)
        value=$(echo "$line" | cut -d=' ' -f2- | xargs)
        # Check if the key exists in the section, if not, add it
        if grep -A10 "^\[DEFAULT\]" "$config_file" | grep -q "^$key ="; then
            echo "Updating $key in DEFAULT section"
            sudo sed -i "/^\[DEFAULT\]/,/^\[/ s/^$key =.*|$key = $value|" "$config_file"
        else
            echo "Adding $key to DEFAULT section"
            sudo sed -i "/^\[DEFAULT\]/a$key = $value" "$config_file"
        fi
    done <<< "$def"
    echo "DEFAULT added to [DEFAULT] section"
else
    echo "[DEFAULT] section not found in $config_file"
    echo "Adding the [DEFAULT] section and new connection string"
    # If the [DEFAULT] section doesn't exist, append it to the config file
    echo "[DEFAULT]" >> "$config_file"
    # Add the DEFAULT below the [DEFAULT] section
    while IFS= read -r line; do
        key=$(echo "$line" | cut -d=' ' -f1 | xargs)
        value=$(echo "$line" | cut -d=' ' -f2- | xargs)
        # Check if the key exists in the section, if not, add it
        if grep -A10 "^\[DEFAULT\]" "$config_file" | grep -q "^$key ="; then
            echo "Updating $key in DEFAULT section"
            sudo sed -i "/^\[DEFAULT\]/,/^\[/ s/^$key =.*|$key = $value|" "$config_file"
        else
            echo "Adding $key to DEFAULT section"
            sudo sed -i "/^\[DEFAULT\]/a$key = $value" "$config_file"
        fi
    done <<< "$def"
fi

```

```

else
    echo "$config_file not found.
    exiting....."
    exit 1
fi

sudo touch /etc/tgt/conf.d/cinder.conf
coff="/etc/tgt/conf.d/cinder.conf"
if [ -f "$coff" ]; then
    echo "include /var/lib/cinder/volumes/*" >> "$coff"
else
    echo "$coff not found"
    exit 1
fi
echo "Storage Node configuration done!!"

# Restart the Block Storage volume service including its dependencies:
sudo service tgt restart
sudo service cinder-volume restart
sudo systemctl restart iscsid

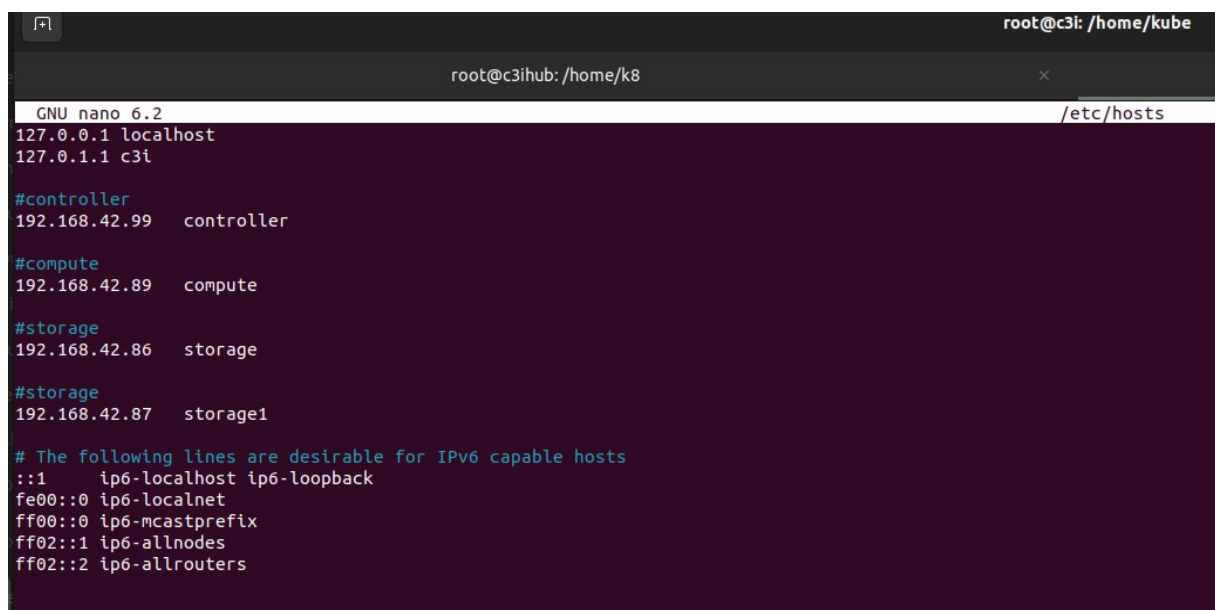
# create the file for nova
sudo touch /var/lib/nova/tmp/os-brick-connect_volume
chown -R nova:nova /var/lib/nova/tmp/

```

Openstack Installation Steps for Compute Node/Storage Node

a) Configure name resolution

- Set the hostname of the node to controller.
- Edit the /etc/hosts file to contain the following:



The screenshot shows a terminal window with the nano text editor open, editing the /etc/hosts file. The terminal title bar indicates the user is root@c3i in the directory /home/kube. The nano editor's status bar shows 'GNU nano 6.2' and the file path '/etc/hosts'. The content of the file is as follows:

```

127.0.0.1 localhost
127.0.1.1 c3i

#controller
192.168.42.99 controller

#compute
192.168.42.89 compute

#storage
192.168.42.86 storage

#storage
192.168.42.87 storage1

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

```

b) Prerequisites Installation

NTP server installation.

```
#!/bin/bash

# update the system
sudo apt update

# Define variables
NTP_SERVER="192.168.42.87" # Replace with your NTP server
SUBNET="192.168.42.87/24" # Replace with your subnet if different

#---chrony---
# install the chrony
sudo apt install chrony -y

# Update chrony configuration
CONFIG_FILE="/etc/chrony/chrony.conf"
echo "Creating the backup of the original file"
cp $CONFIG_FILE $CONFIG_FILE.bak

echo "Updating /etc/chrony/chrony.conf..."
sudo bash -c "cat <<EOF >> $CONFIG_FILE
# Use public servers from the pool.ntp.org project.
server $NTP_SERVER iburst

# Allow clients from the local subnet
allow $SUBNET

# If necessary, replace $SUBNET with a description of your subnet.
EOF"

# Restart chrony service
echo "Restarting chrony service..."
sudo systemctl restart chrony

# verify Operation
echo "Verify Operations"
chronyc sources
```

c) Nova (compute service) installation

```
#!/bin/bash

# Install the packages.
sudo apt install nova-compute -y

metadata_id=$(openssl rand -hex 10 -f value -c id)

# configure the /etc/nova/nova.conf file
config_file="/etc/nova/nova.conf"
trans_url="transport_url = rabbit://openstack:9cada98b1990cc0632d1@controller:5672/"
api="auth_strategy = keystone"
keystone_authtoken_config="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 = nova
password = 5fb4757e4fc20fb39d01"
service_user="send_service_user_token = true
auth_url = https://controller/identity
auth_strategy = keystone
auth_type = password
project_domain_name = Default
project_name = service
user_domain_name = Default
username = nova
password = 5fb4757e4fc20fb39d01"
vnc="enabled = true
server_listen = 0.0.0.0
server_proxyclient_address = 192.168.42.87
novncproxy_base_url = http://controller:6080/vnc_auto.html"
glance="api_servers = http://controller:9292"
oslo_con="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 = 5871a124c5fd28d0ee7f"
neutron="region_name = RegionOne
project_domain_name = Default
project_name = service
auth_type = password
user_domain_name = Default
auth_url = http://controller:5000/v3
username = neutron
password = 2847732ba08d1fca98d4
service_metadata_proxy = true
metadata_proxy_shared_secret = $metadata_id"

if [ -f "$config_file" ]; then
    echo "updating the $config_file"
    sudo cp $config_file $config_file.bak
    # Check if the [DEFAULT] section exists
    if grep -q "^\[DEFAULT\]" "$config_file"; then
        echo "[DEFAULT] section found in $config_file"

        # Check if there is already a transport_url string in the [Default] section
        if grep -A1 "^\[DEFAULT\]" "$config_file" | grep -q "^transport_url ="; then
            echo "Existing transport_url string found, commenting it out"

            # Comment out the existing transport_url string
            sudo sed -i "s/^\.*transport_url = .*/# &/" "$config_file"
        else
            echo "No existing connection string found in the [default] section"
        fi
    fi

```



```

# Add the new connection string below the [database] section
sudo sed -i "/^\[DEFAULT\]/a$trans_url" "$config_file"
echo "New connection string added to [DEFAULT] section"
else
    echo "[DEFAULT] section not found in $config_file"
    echo "Adding the [DEFAULT] section and new connection string"
# If the [DEFAULT] section doesn't exist, append it to the config file
echo -e "\n[DEFAULT]\n$trans_url" | sudo tee -a "$config_file"
#echo -e "\n[DEFAULT]\n$my_ip" | sudo tee -a "$config_file"
fi
# edit the [api] section for keystone
if grep -q "^\[api\]" "$config_file"; then
    echo "[api] section found in $config_file"

    # Add the paste_deploy below the [paste_deploy] section
    sudo sed -i "/^\[api\]/a$api" "$config_file"
    echo "auth_strategy added to [api] section"
else
    echo "[api] section not found in $config_file"
    echo "Adding the [api] section and keystone auth"

# If the [api] section doesn't exist, append it to the config file
echo -e "\n[api]\n$api" | sudo tee -a "$config_file"
fi
# Editing the [keystone_authtoken] section
if grep -q "^\[keystone_authtoken\]" "$config_file"; then
    echo "[keystone_authtoken] section found in $config_file"

    # Add the keystone_authtoken below the [keystone_authtoken] section
    while IFS= read -r line; do
        key=$(echo "$line" | cut -d=' ' -f1 | xargs)
        value=$(echo "$line" | cut -d=' ' -f2- | xargs)

        # Check if the key exists in the section, if not, add it
        if grep -A10 "^\[keystone_authtoken\]" "$config_file" | grep -q "^\$key ="; then
            echo "Updating $key in keystone_authtoken section"
            sudo sed -i "/^\[keystone_authtoken\]/,/^\[/ s/^\$key =.*|\$key = \$value|" "$config_file"
        else
            echo "Adding $key to keystone_authtoken section"
            sudo sed -i "/^\[keystone_authtoken\]/a$key = $value" "$config_file"
        fi
    done <<< "$keystone_authtoken_config"
    echo "keystone_authtoken added to [keystone_authtoken] section"
else
    echo "[keystone_authtoken] section not found in $config_file"
    echo "Adding the [keystone_authtoken] section and new connection string"

# If the [keystone_authtoken] section doesn't exist, append it to the config file
echo "[keystone_authtoken]" >> "$config_file"
# Add the keystone_authtoken below the [keystone_authtoken] section
while IFS= read -r line; do
    key=$(echo "$line" | cut -d=' ' -f1 | xargs)
    value=$(echo "$line" | cut -d=' ' -f2- | xargs)
    # Check if the key exists in the section, if not, add it
    if grep -A10 "^\[keystone_authtoken\]" "$config_file" | grep -q "^\$key ="; then
        echo "Updating $key in keystone_authtoken section"
    else
        echo "Adding $key to keystone_authtoken section"
        sudo sed -i "/^\[keystone_authtoken\]/a$key = $value" "$config_file"
    fi
done <<< "$keystone_authtoken_config"

```

```

        sudo sed -i "/^\[keystone_authtoken\]/,/^\[/ s/^\$key =.*|\$key = \$value|" "$config_file"
    else
        echo "Adding $key to keystone_authtoken section"
        sudo sed -i "/^\[keystone_authtoken\]/a\$key = \$value" "$config_file"
    fi
done <<< "$keystone_authtoken_config"
Fi
# Editing the [service_user] section
if grep -q "^\[service_user\]" "$config_file"; then
    echo "[service_user] section found in $config_file"

    # Add the service_user below the [service_user] section
    while IFS= read -r line; do
        key=$(echo "$line" | cut -d=' ' -f1 | xargs)
        value=$(echo "$line" | cut -d=' ' -f2- | xargs)

        # Check if the key exists in the section, if not, add it
        if grep -A10 "^\[service_user\]" "$config_file" | grep -q "^\$key ="; then
            echo "Updating $key in service_user section"
            sudo sed -i "/^\[service_user\]/,/^\[/ s/^\$key =.*|\$key = \$value|" "$config_file"
        else
            echo "Adding $key to service_user section"
            sudo sed -i "/^\[service_user\]/a\$key = \$value" "$config_file"
        fi
    done <<< "$service_user"
    echo "service_user added to [service_user] section"

else
    echo "[service_user] section not found in $config_file"
    echo "Adding the [service_user] section and new connection string"

# If the [service_user] section doesn't exist, append it to the config file
echo "[service_user]" >> "$config_file"

# Add the service_user below the [service_user] section
while IFS= read -r line; do
    key=$(echo "$line" | cut -d=' ' -f1 | xargs)
    value=$(echo "$line" | cut -d=' ' -f2- | xargs)
    # Check if the key exists in the section, if not, add it
    if grep -A10 "^\[service_user\]" "$config_file" | grep -q "^\$key ="; then
        echo "Updating $key in service_user section"
        sudo sed -i "/^\[service_user\]/,/^\[/ s/^\$key =.*|\$key = \$value|" "$config_file"
    else
        echo "Adding $key to service_user section"
        sudo sed -i "/^\[service_user\]/a\$key = \$value" "$config_file"
    fi
done <<< "$service_user"
fi

# edit the [vnc] section
if grep -q "^\[vnc\]" "$config_file"; then
    echo "[vnc] section found in $config_file"

    # Add the vnc below the [vnc] section
    while IFS= read -r line; do
        key=$(echo "$line" | cut -d=' ' -f1 | xargs)
        value=$(echo "$line" | cut -d=' ' -f2- | xargs)

```

```

# Check if the key exists in the section, if not, add it
if grep -A10 "^\[vnc\]" "$config_file" | grep -q "^$key ="; then
    echo "Updating $key in vnc section"
    sudo sed -i "/^\[vnc\]/,/^\[/ s|^$key =.*|$key = $value|" "$config_file"
else
    echo "Adding $key to vnc section"
    sudo sed -i "/^\[vnc\]/a$key = $value" "$config_file"
fi
done <<< "$vnc"
echo "service_user added to [vnc] section"
else
    echo "[vnc] section not found in $config_file"
    echo "Adding the [vnc] section and new connection string"
# If the [vnc] section doesn't exist, append it to the config file
echo "[vnc]" >> "$config_file"
# Add the vnc below the [vnc] section
while IFS= read -r line; do
    key=$(echo "$line" | cut -d=' ' -f1 | xargs)
    value=$(echo "$line" | cut -d=' ' -f2- | xargs)
    # Check if the key exists in the section, if not, add it
    if grep -A10 "^\[vnc\]" "$config_file" | grep -q "^$key ="; then
        echo "Updating $key in vnc section"
        sudo sed -i "/^\[vnc\]/,/^\[/ s|^$key =.*|$key = $value|" "$config_file"
    else
        echo "Adding $key to vnc section"
        sudo sed -i "/^\[vnc\]/a$key = $value" "$config_file"
    fi
done <<< "$vnc"
fi
# edit glance section
if grep -q "^\[glance\]" "$config_file"; then
    echo "[glance] section found in $config_file"
    # Add the api_server below the [glance] section
    sudo sed -i "/^\[glance\]/a$glance" "$config_file"
    echo "glance added to [glance] section"
else
    echo "[glance] section not found in $config_file"
    echo "Adding the [glance] section and new connection string"
# If the [glance] section doesn't exist, append it to the config file
echo -e "\n[glance]\n$glance" | sudo tee -a "$config_file"
fi
# edit oslo_concurrency section
if grep -q "^\[oslo_concurrency\]" "$config_file"; then
    echo "[oslo_concurrency] section found in $config_file"
    # Add the lock_path below the [oslo_concurrency] section
    sudo sed -i "/^\[oslo_concurrency\]/a$oslo_con" "$config_file"
    echo "lock_path added to [oslo_concurrency] section"
else
    echo "[oslo_concurrency] section not found in $config_file"
    echo "Adding the [oslo_concurrency] section and new connection string"
# If the [oslo_concurrency] section doesn't exist, append it to the config file
echo -e "\n[oslo_concurrency]\n$oslo_con" | sudo tee -a "$config_file"
fi
# edit the [placement] section
if grep -q "^\[placement\]" "$config_file"; then

```

```

echo "[placement] section found in $config_file"
# Add the placement-conf below the [placement] section
while IFS= read -r line; do
    key=$(echo "$line" | cut -d=' ' -f1 | xargs)
    value=$(echo "$line" | cut -d=' ' -f2- | xargs)
# Check if the key exists in the section, if not, add it
    if grep -A10 "\[placement\]" "$config_file" | grep -q "^$key ="; then
        echo "Updating $key in placement section"
        sudo sed -i "/\[placement\]/,/[/ s/^\$key =.*|\$key = \$value|" "$config_file"
    else
        echo "Adding $key to placement section"
        sudo sed -i "/\[placement\]/a\$key = \$value" "$config_file"
    fi
done <<< "$placement"
echo "placement-conf added to [placement] section"
else
    echo "[placement] section not found in $config_file"
    echo "Adding the [placement] section and new connection string"
# If the [placement] section doesn't exist, append it to the config file
echo -e "\n[placement]\n\$oslo_con" | sudo tee -a "$config_file"
fi
# edit the [neutron] section
if grep -q "\[neutron\]" "$config_file"; then
    echo "[neutron] section found in $config_file"
    # Add the neutron-conf below the [neutron] section
    while IFS= read -r line; do
        key=$(echo "$line" | cut -d=' ' -f1 | xargs)
        value=$(echo "$line" | cut -d=' ' -f2- | xargs)
        # Check if the key exists in the section, if not, add it
        if grep -A10 "\[neutron\]" "$config_file" | grep -q "^$key ="; then
            echo "Updating $key in neutron section"
            sudo sed -i "/\[neutron\]/,/[/ s/^\$key =.*|\$key = \$value|" "$config_file"
        Else
            echo "Adding $key to neutron section"
            sudo sed -i "/\[neutron\]/a\$key = \$value" "$config_file"
        fi
    done <<< "$neutron"
    echo "neutron-conf added to [neutron] section"
else
    echo "[neutron] section not found in $config_file"
    echo "Adding the [neutron] section and new connection string"

# If the [neutron] section doesn't exist, append it to the config file
echo -e "\n[neutron]\n\$oslo_con" | sudo tee -a "$config_file"
# Add the neutron-conf below the [neutron] section
while IFS= read -r line; do
    key=$(echo "$line" | cut -d=' ' -f1 | xargs)
    value=$(echo "$line" | cut -d=' ' -f2- | xargs)
    # Check if the key exists in the section, if not, add it
    if grep -A10 "\[neutron\]" "$config_file" | grep -q "^$key ="; then
        echo "Updating $key in neutron section"
        sudo sed -i "/\[neutron\]/,/[/ s/^\$key =.*|\$key = \$value|" "$config_file"
    else
        echo "Adding $key to neutron section"
        sudo sed -i "/\[neutron\]/a\$key = \$value" "$config_file"
    fi

```

```

done <<< "$neutron"
echo "neutron-conf added to [neutron] section"
fi

else
echo "$config_file not found.
exiting....."
exit 1
fi

val=$(egrep -c '(vmx|svm)' /proc/cpuinfo)

if [ $val -eq 0 ]; then
config_file1="/etc/nova/nova-compute.conf"
libvirt="virt_type = qemu"
if [ -f "$config_file1" ]; then
echo "updating the $config_file1"
sudo cp $config_file1 $config_file2.bak

# edit libvirt section
if grep -q "^\[libvirt\]" "$config_file1"; then
echo "[libvirt] section found in $config_file1"

# Add the lock_path below the [libvirt] section
sudo sed -i "/^\[libvirt\]/a$libvirt" "$config_file1"
echo "lock_path added to [libvirt] section"

else
echo "[libvirt] section not found in $config_file1"
echo "Adding the [libvirt] section and new connection string"

# If the [libvirt] section doesn't exist, append it to the config file
echo -e "\n\[libvirt\]\n$libvirt" | sudo tee -a "$config_file1"
fi

else
echo "$config_file1 not found.
exiting....."
exit 1
fi

else
echo "using kvm hypervisor"
fi

sudo chown nova:nova -R /etc/nova/
sudo chown nova:nova -R /etc/nova/nova.conf
sudo chmod -R 755 /etc/nova/nova.conf
sudo chown nova:nova -R /var/lib/nova/instances/
sudo chmod -R 755 /var/lib/nova/instances/

#restart the service
service nova-compute restart

```

c) Neutron (Networking service) installation

```
#!/bin/bash

#---- Neutron Installation-----
# install the neutron packages.
sudo apt install neutron-openvswitch-agent -y
# Edit the /etc/neutron/neutron.conf file and complete the following actions. In the [database] sec>

config_file="/etc/neutron/neutron.conf"
def_trans="transport_url = rabbit://openstack:9cada98b1990cc0632d1@controller:5672/"
oslo_conf="lock_path = /var/lib/neutron/tmp"

config_file2="/etc/neutron/plugins/ml2/openvswitch_agent.ini"
ovs="bridge_mappings = provider:br-ex
local_ip = 192.168.42.87"
agent="tunnel_types = vxlan
l2_population = true"
securitygroup="enable_security_group = true
firewall_driver = openvswitch"

if [ -f "$config_file" ]; then
    echo "updating the $config_file"
    sudo cp $config_file $config_file.bak

    # Check if the [DEFAULT] section exists
    if grep -q "\[DEFAULT\]" "$config_file"; then
        echo "[DEFAULT] section found in $config_file"

        # Check if there is already a transport_url string in the [Default] section
        if grep -A1 "\[DEFAULT\]" "$config_file" | grep -q "^transport_url ="; then
            echo "Existing transport_url string found, commenting it out"
            # Comment out the existing transport_url string
            sudo sed -i "s/^.*transport_url = .*/# &/" "$config_file"
        else
            echo "No existing connection string found in the [default] section"
        fi

        # Add the new connection string below the [database] section
        sudo sed -i "\[DEFAULT\]/a$def_trans" "$config_file"
        echo "New connection string added to [DEFAULT] section"
    Else
        echo "[DEFAULT] section not found in $config_file"
        echo "Adding the [DEFAULT] section and new connection string"

        # If the [DEFAULT] section doesn't exist, append it to the config file
        echo -e "\n[DEFAULT]\n\n$def_trans" | sudo tee -a "$config_file"
    fi

    # edit oslo_concurrency section
    if grep -q "\[oslo_concurrency\]" "$config_file"; then
        echo "[oslo_concurrency] section found in $config_file"

        # Add the api_server below the [glance] section
        sudo sed -i "\[oslo_concurrency\]/a$oslo_conf" "$config_file"
        echo "oslo_concurrency added to [oslo_concurrency] section"
    fi
fi
```

```

else
    echo "[oslo_concurrency] section not found in $config_file"
    echo "Adding the [oslo_concurrency] section and new connection string"

    # If the [oslo_concurrency] section doesn't exist, append it to the config file
    echo -e "\n[oslo_concurrency]\n$oslo_conf" | sudo tee -a "$config_file"
fi

else
    echo "$config_file not found.
    exiting....."
    exit 1
fi

# configure the $config_file2. check that file was found or not
if [ -f "$config_file2" ]; then
    echo "updating the $config_file2"
    sudo cp $config_file2 $config_file2.bak

    # Editing the [ovs] section
    if grep -q "^\[ovs\]" "$config_file2"; then
        echo "[ovs] section found in $config_file2"

        # Add the nova config below in the [ovs] section
        while IFS= read -r line; do
            key=$(echo "$line" | cut -d=' ' -f1 | xargs)
            value=$(echo "$line" | cut -d=' ' -f2- | xargs)

            # Check if the key exists in the section, if not, add it
            if grep -A10 "^\[ovs\]" "$config_file2" | grep -q "^\$key ="; then
                echo "Updating $key in ovs section"
                sudo sed -i "/^\[ovs\]/,/^\[ / s/^\$key =.*|\$key = \$value|" "$config_file2"
            else
                echo "Adding $key to ovs section"
                sudo sed -i "/^\[ovs\]/a\$key = \$value" "$config_file2"
            fi
        done <<< "$ovs"
        echo "ovs configuration added to [ovs] section"

    else
        echo "[ovs] section not found in $config_file2"
        echo "Adding the [ovs] section and new connection string"
        # If the [ovs] section doesn't exist, append it to the config file
        echo "[ovs]" >> "$config_file2"
        # Add the ovs configuration below the [ovs] section
        while IFS= read -r line; do
            key=$(echo "$line" | cut -d=' ' -f1 | xargs)
            value=$(echo "$line" | cut -d=' ' -f2- | xargs)
            # Check if the key exists in the section, if not, add it
            if grep -A10 "^\[ovs\]" "$config_file2" | grep -q "^\$key ="; then
                echo "Updating $key in ovs section"
                sudo sed -i "/^\[ovs\]/,/^\[ / s/^\$key =.*|\$key = \$value|" "$config_file2"
            else
                echo "Adding $key to ovs section"
                sudo sed -i "/^\[ovs\]/a\$key = \$value" "$config_file2"
            fi
        done <<< "$ovs"
    fi
fi

```

```

# Editing the [agent] section
if grep -q "^\[agent\]" "$config_file2"; then
    echo "[agent] section found in $config_file2"

    # Add the agent config below in the [agent] section
    while IFS= read -r line; do
        key=$(echo "$line" | cut -d=' ' -f1 | xargs)
        value=$(echo "$line" | cut -d=' ' -f2- | xargs)
        # Check if the key exists in the section, if not, add it
        if grep -A10 "^\[agent\]" "$config_file2" | grep -q "^$key ="; then
            echo "Updating $key in agent section"
            sudo sed -i "/^\[agent\]/,/^\[ / s/^$key =.*$key = $value|" "$config_file2"
        else
            echo "Adding $key to agent section"
            sudo sed -i "/^\[agent\]/a$key = $value" "$config_file2"
        fi
    done <<< "$agent"
    echo "agent configuration added to [agent] section"

else
    echo "[agent] section not found in $config_file2"
    echo "Adding the [agent] section and new connection string"

# If the [agent] section doesn't exist, append it to the config file
echo "[agent]" >> "$config_file2"

# Add the agent configuration below the [agent] section
while IFS= read -r line; do
    key=$(echo "$line" | cut -d=' ' -f1 | xargs)
    value=$(echo "$line" | cut -d=' ' -f2- | xargs)
    # Check if the key exists in the section, if not, add it
    if grep -A10 "^\[agent\]" "$config_file2" | grep -q "^$key ="; then
        echo "Updating $key in agent section"
        sudo sed -i "/^\[agent\]/,/^\[ / s/^$key =.*$key = $value|" "$config_file2"
    else
        echo "Adding $key to agent section"
        sudo sed -i "/^\[agent\]/a$key = $value" "$config_file2"
    fi
done <<< "$agent "
fi

# Editing the [securitygroup] section
if grep -q "^\[securitygroup\]" "$config_file2"; then
    echo "[securitygroup] section found in $config_file2"

    # Add the agent config below in the [agent] section
    while IFS= read -r line; do
        key=$(echo "$line" | cut -d=' ' -f1 | xargs)
        value=$(echo "$line" | cut -d=' ' -f2- | xargs)

        # Check if the key exists in the section, if not, add it
        if grep -A10 "^\[securitygroup\]" "$config_file2" | grep -q "^$key ="; then
            echo "Updating $key in securitygroup section"
            sudo sed -i "/^\[securitygroup\]/,/^\[ / s/^$key =.*$key = $value|" "$config_file2"
        else
            echo "Adding $key to securitygroup section"
            sudo sed -i "/^\[securitygroup\]/a$key = $value" "$config_file2"
        fi
    done <<< "$agent "
fi

```



```

done <<< "$securitygroup"
echo "securitygroup configuration added to [securitygroup] section"

else
echo "[securitygroup] section not found in $config_file2"
echo "Adding the [securitygroup] section and new connection string"

# If the [securitygroup] section doesn't exist, append it to the config file
echo "[securitygroup]" >> "$config_file2"

# Add the securitygroup configuration below the [securitygroup] section
while IFS= read -r line; do
key=$(echo "$line" | cut -d=' ' -f1 | xargs)
value=$(echo "$line" | cut -d=' ' -f2- | xargs)
# Check if the key exists in the section, if not, add it
if grep -A10 "^\[securitygroup\]" "$config_file2" | grep -q "^\$key ="; then
echo "Updating $key in securitygroup section"
sudo sed -i "/^\[securitygroup\]/,/^\[/ s/^\$key =.*|\$key = $value|" "$config_file2"
else
echo "Adding $key to securitygroup section"
sudo sed -i "/^\[securitygroup\]/a\$key = $value" "$config_file2"
fi
done <<< "$securitygroup"
fi

sudo systemctl net.bridge.bridge-nf-call-iptables = 1
sudo systemctl net.bridge.bridge-nf-call-ip6tables = 1
#sudo ovs-vsctl add-br br-ex
#sudo ovs-vsctl add-port br-ex enp0s8
#sudo ifconfig br-ex up
#sudo ip addr flush dev enp0s8
#sudo ip addr add 172.29.24.161/22 dev br-ex
#sudo ip route add default via 172.29.27.254 dev br-ex proto dhcp metric 100
#sudo ip link set br-ex
echo "conf done!!"

else
echo "$config_file2 not found.
exiting....."
exit 1
fi

# service restart
service neutron-openvswitch-agent restart

```

c) Cinder (Volume service) installation

```
#!/bin/bash

# install supporting utility packages.
sudo apt install lvm2 thin-provisioning-tools -y

# Create the LVM physical volume /dev/sdb:
sudo pvcreate /dev/sda3

# Create the LVM volume group cinder-volumes:
sudo vgcreate cinder-volumes /dev/sda3

conf="/etc/lvm/lvm.conf"
sudo cp $conf $conf.bak
sed -i '/devices {/!b;n;c\    filter = [ "a/sda3/", "r./.*/" ]' $conf
echo "lvm conf done...."

#Install the packages:
sudo apt install cinder-volume tgt -y

# Edit the /etc/cinder/cinder.conf file and complete the following actions:
config_file="/etc/cinder/cinder.conf"
db_con="connection = mysql+pymysql://cinder:d4a777d18ba37a0cb402@controller/cinder"
trans_url="transport_url = rabbit://openstack:9cada98b1990cc0632d1@controller"
def_ser="auth_strategy = keystone"
keystone_authtoken_config="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 = cinder
password = d4a777d18ba37a0cb402
service_token_roles_required = True
service_token_roles = admin"
oslo_con="lock_path = /var/lib/cinder/tmp"
lvm="volume_driver = cinder.volume.drivers.lvm.LVMVolumeDriver
volume_group = cinder-volumes
target_protocol = iscsi
target_helper = tgtadm"
def="enabled_backends = lvm
glance_api_servers = http://controller:9292"
if [ -f "$config_file" ]; then
    echo "updating the $config_file"
    sudo cp $config_file $config_file.bak
    sudo chown cinder:cinder /etc/cinder/cinder.conf
    #In the [database] and [database] sections, configure database access.
    if grep -q "^\[database\]" "$config_file"; then
        echo "[database] section found in $config_file"
        # Check if there is already a connection string in the [database] section
        if grep -A1 "^\[database\]" "$config_file" | grep -q "^connection ="; then
            echo "Existing connection string found, commenting it out"
            # Comment out the existing connection string
            sudo sed -i 's/^.*connection = .*/# &/' "$config_file"
```

```

else
    echo "No existing connection string found in the [database] section"
fi
# Add the new connection string below the [database] section
sudo sed -i "/^\[database\]/a$db_con" "$config_file"
echo "New connection string added to [database] section"
else
    echo "[database] section not found in $config_file"
    echo "Adding the [database] section and new connection string"
# If the [database] section doesn't exist, append it to the config file
echo -e "\n[database]\n$db_con" | sudo tee -a "$config_file"
fi

#In the [DEFAULT] section, configure RabbitMQ message queue access.
if grep -q "^\[DEFAULT\]" "$config_file"; then
    echo "[DEFAULT] section found in $config_file"

    # Check if there is already a transport_url string in the [Default] section
    if grep -A1 "^\[DEFAULT\]" "$config_file" | grep -q "^\transport_url ="; then
        echo "Existing transport_url string found, commenting it out"
        # Comment out the existing transport_url string
        sudo sed -i "s/^\.*transport_url = .*/# &/" "$config_file"
    else
        echo "No existing connection string found in the [default] section"
    fi
    # Add the new transport_url string below the [Default] section
sudo sed -i "/^\[DEFAULT\]/a$trans_url" "$config_file"
    echo "New connection string added to [DEFAULT] section"

else
    echo "[DEFAULT] section not found in $config_file"
    echo "Adding the [DEFAULT] section and new connection string"

# If the [DEFAULT] section doesn't exist, append it to the config file
echo -e "\n[DEFAULT]\n\n$trans_url" | sudo tee -a "$config_file"
fi

# edit the [DEFAULT] section for keystone
if grep -q "^\[DEFAULT\]" "$config_file"; then
    echo "[DEFAULT] section found in $config_file"

    # Add the DEFAULT below the [DEFAULT] section
    sudo sed -i "/^\[DEFAULT\]/a$def_ser" "$config_file"
    echo "auth_strategy added to [DEFAULT] section"

else
    echo "[DEFAULT] section not found in $config_file"
    echo "Adding the [DEFAULT] section and keystone auth"

# If the [DEFAULT] section doesn't exist, append it to the config file
echo -e "\n[DEFAULT]\n\n$def_ser" | sudo tee -a "$config_file"
fi

# Editing the [keystone_authtoken] section
if grep -q "^\[keystone_authtoken\]" "$config_file"; then
    echo "[keystone_authtoken] section found in $config_file"

```

```

# Add the keystone_auth token below the [keystone_auth token] section
while IFS= read -r line; do
    key=$(echo "$line" | cut -d=' ' -f1 | xargs)
    value=$(echo "$line" | cut -d=' ' -f2- | xargs)

    # Check if the key exists in the section, if not, add it
    if grep -A10 "^\[keystone_auth token\]" "$config_file" | grep -q "^\$key ="; then
        echo "Updating $key in keystone_auth token section"
        sudo sed -i "/^\[keystone_auth token\]/,/^\[/ s/^\$key =.*|\$key = \$value|" "$config_file"
    else
        echo "Adding $key to keystone_auth token section"
        sudo sed -i "/^\[keystone_auth token\]/a\$key = \$value" "$config_file"
    fi
done <<< "$keystone_auth token_config"
echo "keystone_auth token added to [keystone_auth token] section"

else
    echo "[keystone_auth token] section not found in $config_file"
    echo "Adding the [keystone_auth token] section and new connection string"

# If the [keystone_auth token] section doesn't exist, append it to the config file
echo "[keystone_auth token]" >> "$config_file"

# Add the keystone_auth token below the [keystone_auth token] section
while IFS= read -r line; do
    key=$(echo "$line" | cut -d=' ' -f1 | xargs)
    value=$(echo "$line" | cut -d=' ' -f2- | xargs)
    # Check if the key exists in the section, if not, add it
    if grep -A10 "^\[keystone_auth token\]" "$config_file" | grep -q "^\$key ="; then
        echo "Updating $key in keystone_auth token section"
        sudo sed -i "/^\[keystone_auth token\]/,/^\[/ s/^\$key =.*|\$key = \$value|" "$config_file"
    else
        echo "Adding $key to keystone_auth token section"
        sudo sed -i "/^\[keystone_auth token\]/a\$key = \$value" "$config_file"
    fi
done <<< "$keystone_auth token_config"
Fi

# edit oslo_concurrency section
if grep -q "^\[oslo_concurrency\]" "$config_file"; then
    echo "[oslo_concurrency] section found in $config_file"

    # Add the lock_path below the [oslo_concurrency] section
    sudo sed -i "/^\[oslo_concurrency\]/a\$oslo_con" "$config_file"
    echo "lock_path added to [oslo_concurrency] section"

else
    echo "[oslo_concurrency] section not found in $config_file"
    echo "Adding the [oslo_concurrency] section and new connection string"

# If the [oslo_concurrency] section doesn't exist, append it to the config file
echo -e "\n[oslo_concurrency]\n\$oslo_con" | sudo tee -a "$config_file"
fi

# Editing the [lvm] section
if grep -q "^\[lvm\]" "$config_file"; then
    echo "[lvm] section found in $config_file"

```

```

# Add the lvm below the [lvm] section
while IFS= read -r line; do
    key=$(echo "$line" | cut -d=' ' -f1 | xargs)
    value=$(echo "$line" | cut -d=' ' -f2- | xargs)

    # Check if the key exists in the section, if not, add it
    if grep -A10 "\[lvm\]" "$config_file" | grep -q "^$key ="; then
        echo "Updating $key in lvm section"
        sudo sed -i "/\[lvm\]/,/^\[ / s/^$key =.*|$key = $value|" "$config_file"
    else
        echo "Adding $key to lvm section"
        sudo sed -i "/\[lvm\]/a$key = $value" "$config_file"
    fi
done <<< "$lvm"
echo "lvm added to [lvm] section"
else
    echo "[lvm] section not found in $config_file"
    echo "Adding the [lvm] section and new connection string"
# If the [lvm] section doesn't exist, append it to the config file
echo "[lvm]" >> "$config_file"
# Add the lvm below the [lvm] section
while IFS= read -r line; do
    key=$(echo "$line" | cut -d=' ' -f1 | xargs)
    value=$(echo "$line" | cut -d=' ' -f2- | xargs)
    # Check if the key exists in the section, if not, add it
    if grep -A10 "\[lvm\]" "$config_file" | grep -q "^$key ="; then
        echo "Updating $key in lvm section"
        sudo sed -i "/\[lvm\]/,/^\[ / s/^$key =.*|$key = $value|" "$config_file"
    else
        echo "Adding $key to lvm section"
        sudo sed -i "/\[lvm\]/a$key = $value" "$config_file"
    fi
done <<< "$lvm"
fi
# Editing the [DEFAULT] section
if grep -q "\[DEFAULT\]" "$config_file"; then
    echo "[DEFAULT] section found in $config_file"
    # Add the lvm below the [DEFAULT] section
    while IFS= read -r line; do
        key=$(echo "$line" | cut -d=' ' -f1 | xargs)
        value=$(echo "$line" | cut -d=' ' -f2- | xargs)
        # Check if the key exists in the section, if not, add it
        if grep -A10 "\[DEFAULT\]" "$config_file" | grep -q "^$key ="; then
            echo "Updating $key in DEFAULT section"
            sudo sed -i "/\[DEFAULT\]/,/^\[ / s/^$key =.*|$key = $value|" "$config_file"
        else
            echo "Adding $key to DEFAULT section"
            sudo sed -i "/\[DEFAULT\]/a$key = $value" "$config_file"
        fi
    done <<< "$def"
    echo "DEFAULT added to [DEFAULT] section"
else
    echo "[DEFAULT] section not found in $config_file"
    echo "Adding the [DEFAULT] section and new connection string"
# If the [DEFAULT] section doesn't exist, append it to the config file
echo "[DEFAULT]" >> "$config_file"

```

```

# Add the DEFAULT below the [DEFAULT] section
while IFS= read -r line; do
    key=$(echo "$line" | cut -d'=' -f1 | xargs)
    value=$(echo "$line" | cut -d'=' -f2- | xargs)
    # Check if the key exists in the section, if not, add it
    if grep -A10 "^\[DEFAULT\]" "$config_file" | grep -q "^$key ="; then
        echo "Updating $key in DEFAULT section"
        sudo sed -i "/^\[DEFAULT\]/,/^\[/ s|^$key =.*|$key = $value|" "$config_file"
    else
        echo "Adding $key to DEFAULT section"
        sudo sed -i "/^\[DEFAULT\]/a$key = $value" "$config_file"
    fi
done <<< "$def"
fi

else
    echo "$config_file not found."
    exiting....."
    exit 1
fi

sudo touch /etc/tgt/conf.d/cinder.conf
coff="/etc/tgt/conf.d/cinder.conf"
if [ -f "$coff" ]; then
    echo "include /var/lib/cinder/volumes/*" >> "$coff"
else
    echo "$coff not found"
    exit 1
fi

echo "Storage Node configuration done!!"

# Restart the Block Storage volume service including its dependencies:
sudo service tgt restart
sudo service cinder-volume restart
sudo systemctl restart iscsid

# create the file for nova
sudo touch /var/lib/nova/tmp/os-brick-connect_volume
chown -R nova:nova /var/lib/nova/tmp/

```

Network Configuration script

Script is used to create the external brige and provide the functionality of the physical interface of the server. This script is also provided in the neutron installation file but needs some chnages.

```
#!/bin/bash

sudo ovs-vsctl add-br br-ex1
sudo ovs-vsctl add-port br-ex1 eno12419 # change the name of the physical interface
sudo ifconfig br-ex1 up
sudo ip addr add 192.168.42.99/24 dev br-ex1 # change the ip address
sudo ip addr flush dev eno12419
#sudo ip link set br-ex
sudo ip route add default via 192.168.42.2 dev br-ex1 proto dhcp metric 100 # change the router address
sudo ip link set br-ex1
```

Screenshots

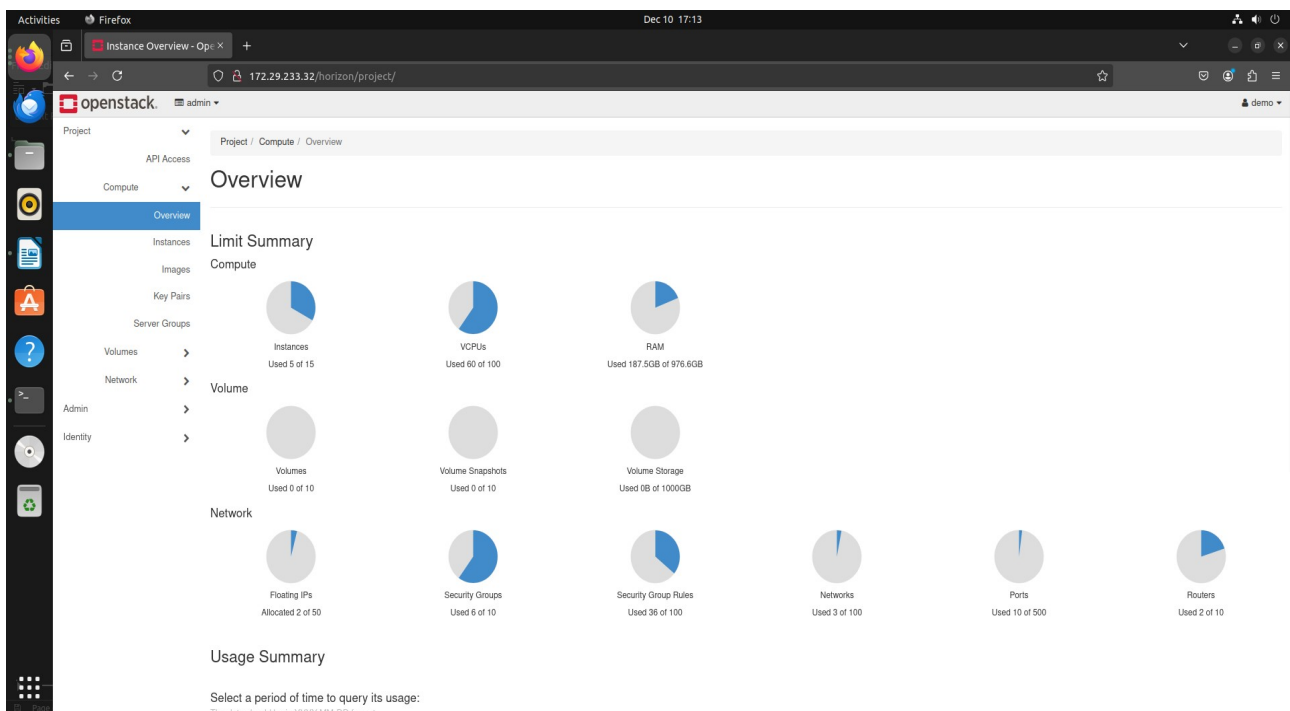


Figure 1: Overview Page

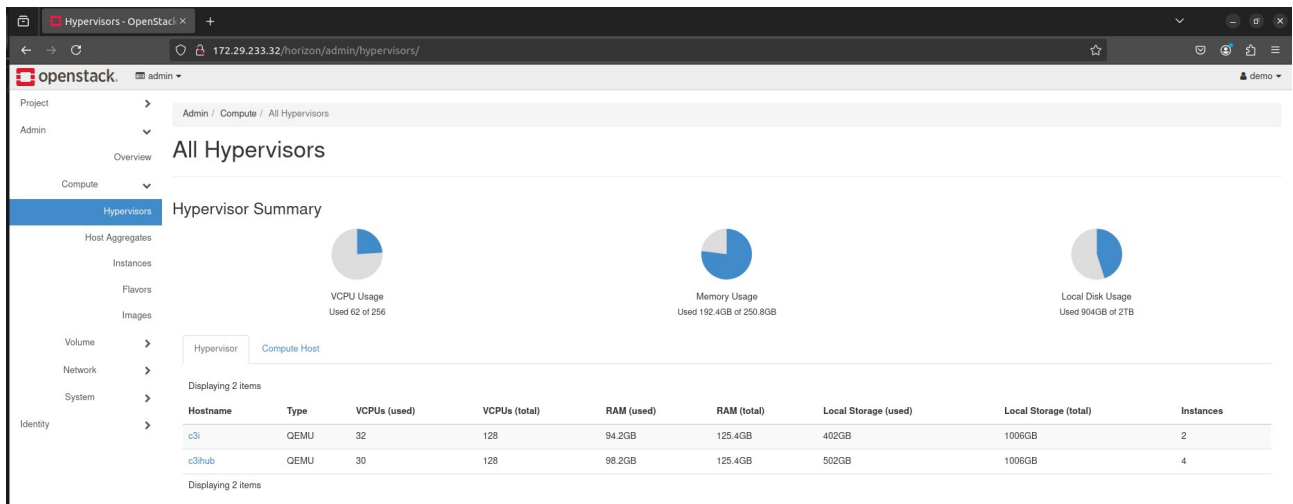


Figure 2: Hypervisor Page

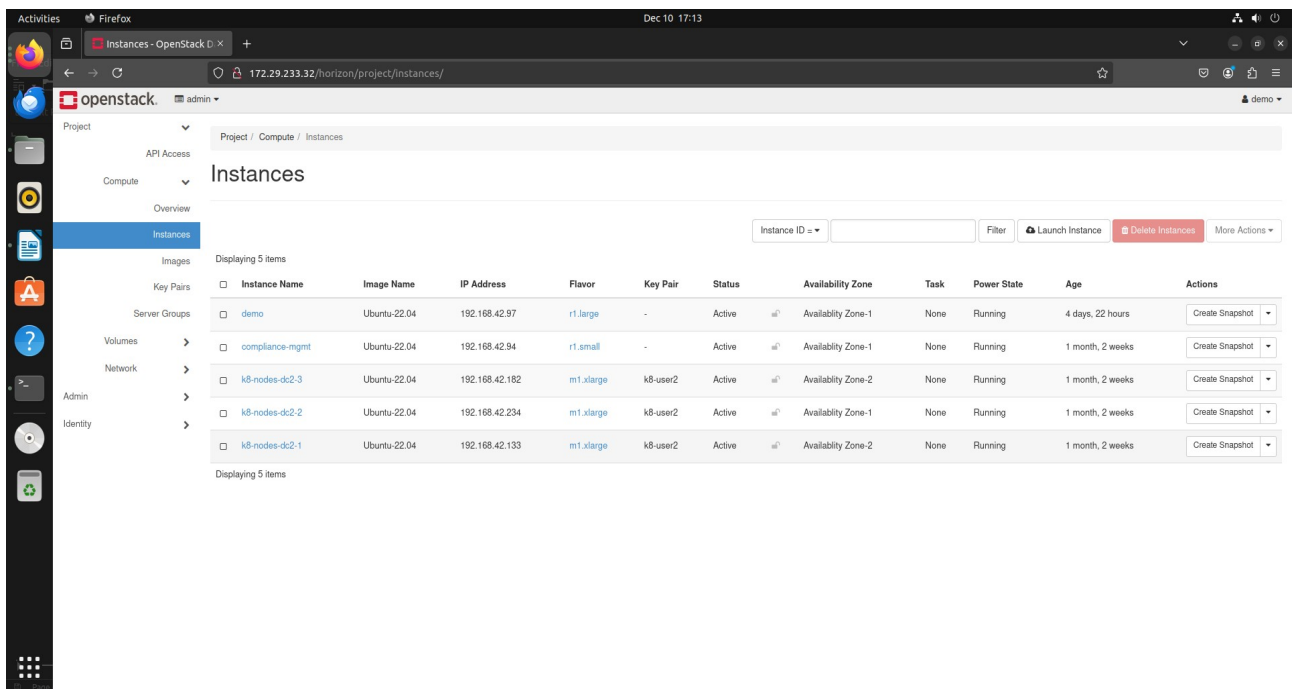


Figure 3: Instances Page

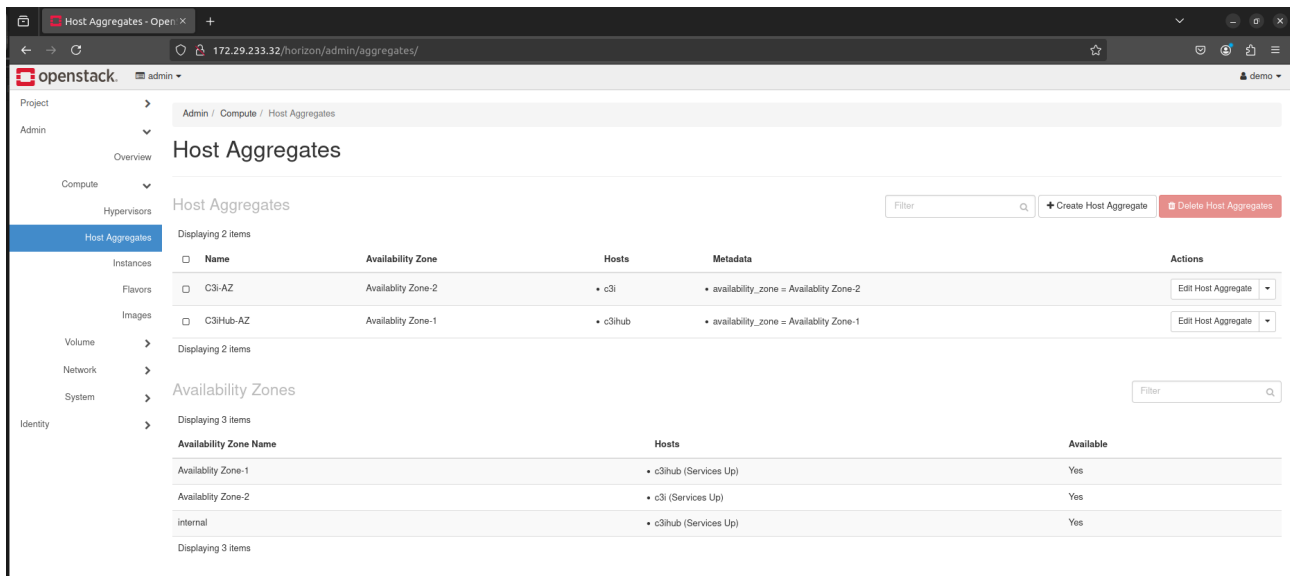


Figure 4: Host Aggregates Page

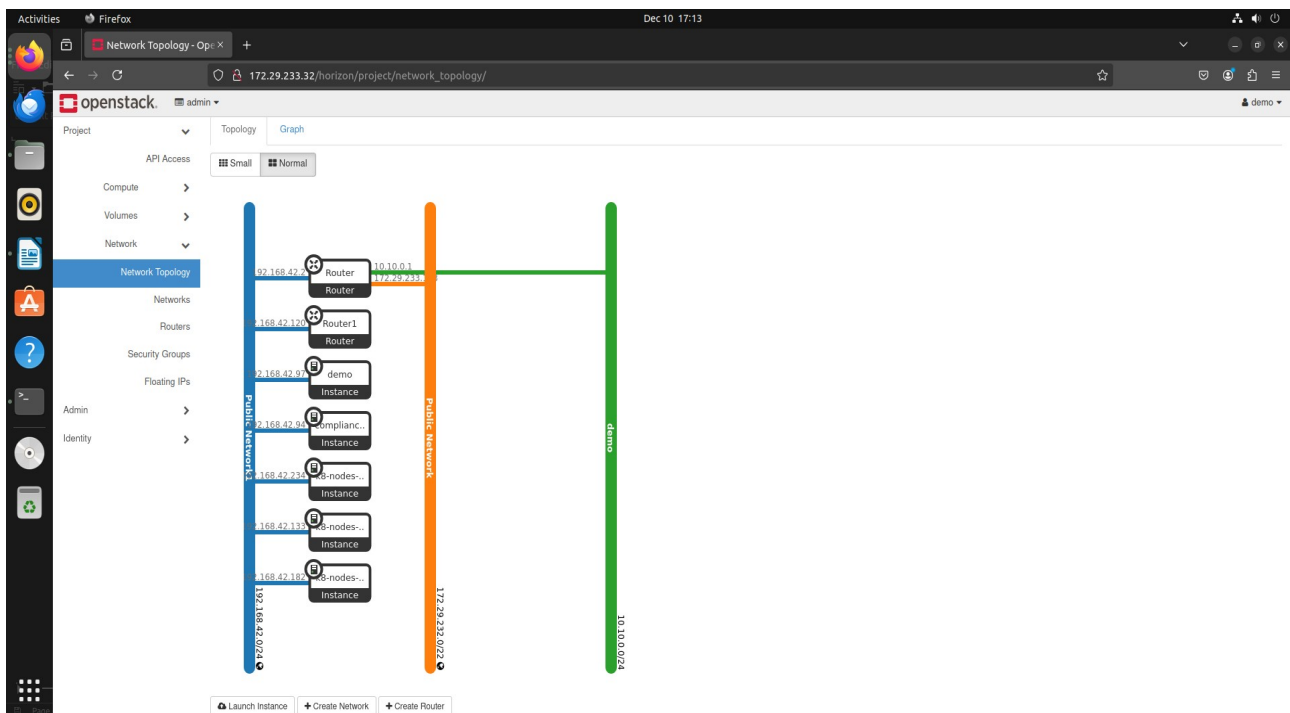


Figure 5: Network Topology Page

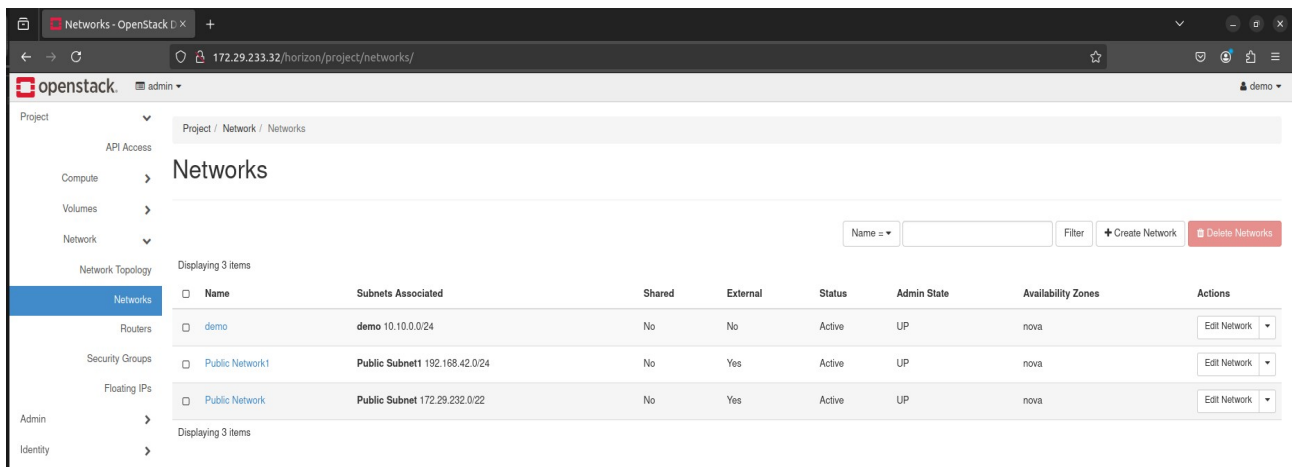


Figure 6: Networks Page

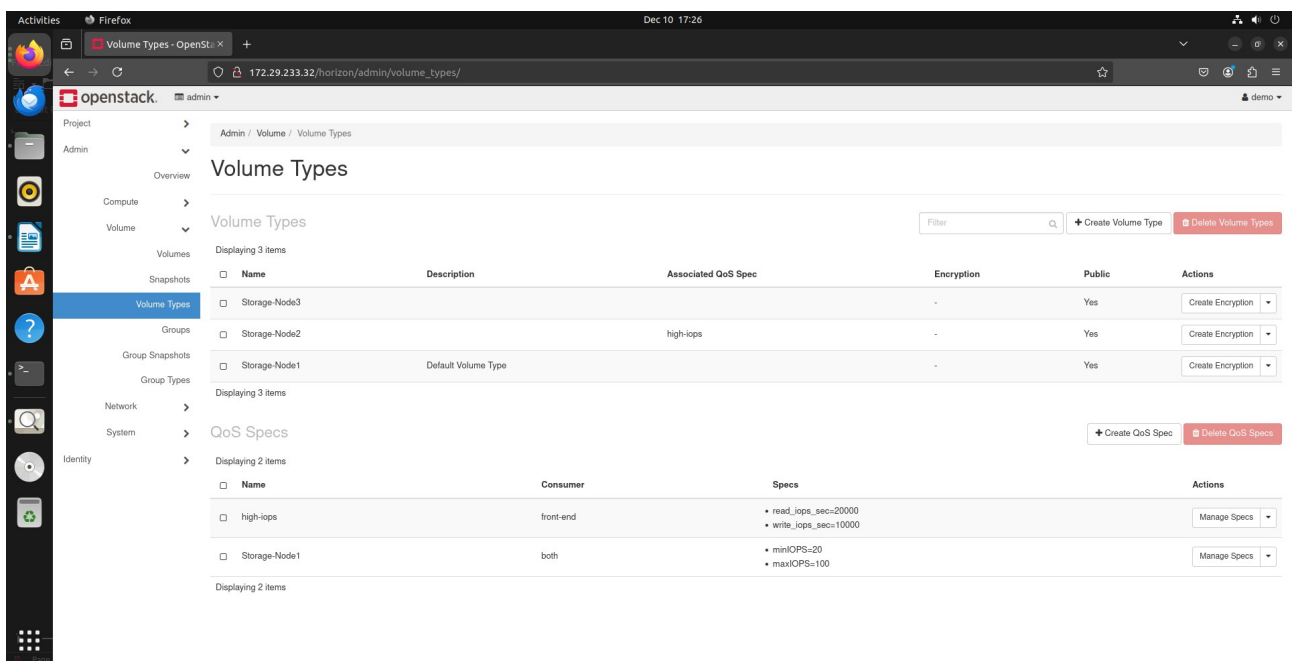


Figure 7: Volume Types Page

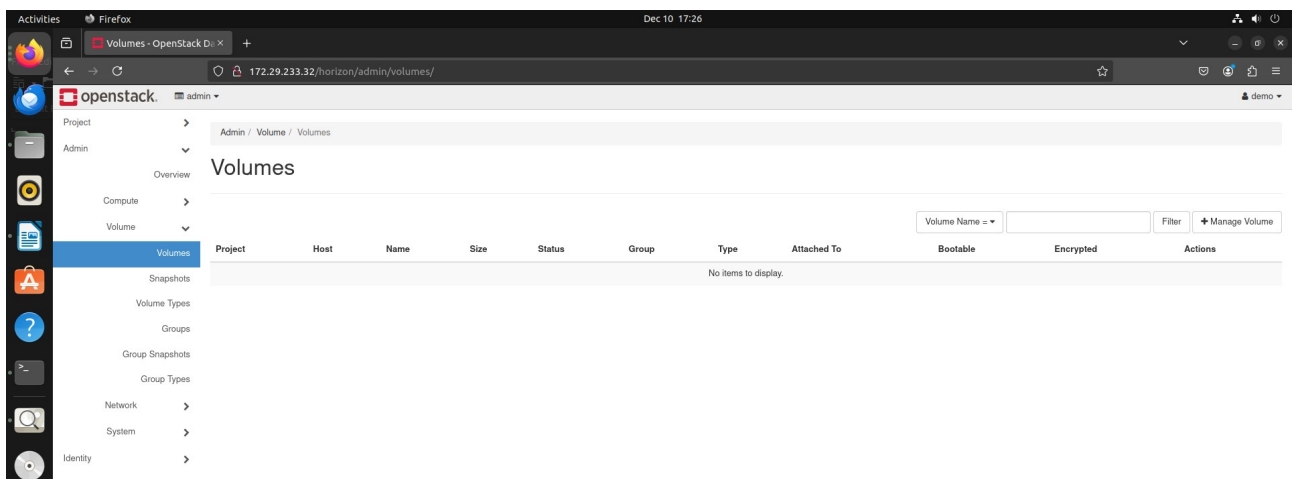


Figure 8: Volume Page

Backup of Openstack Setup

Database Backups

The example OpenStack architecture designates the cloud controller as the MySQL server. This MySQL server hosts the databases for nova, glance, cinder, and keystone. With all of these databases in one place, it's very easy to create a database backup:

```
mysqldump --opt --all-databases > openstack.sql
```

File System Backups

1. Compute

The /etc/nova directory on both the cloud controller and compute nodes should be regularly backed up.

/var/log/nova contains all the logs. So, this directory on both controller and compute nodes should be regularly backed up.

/var/lib/nova is another important directory to back up. The exception to this is the /var/lib/nova/instances subdirectory on compute nodes. This subdirectory contains the KVM images of running instances. If need to maintain backup copies of all instances then only back up this directory.

2. Image Catalog and Delivery

/etc/glance and /var/log/glance follow the same rules as their nova counterparts.

/var/lib/glance should also be backed up. Take special notice of /var/lib/glance/images. If using a file-based back end of glance, /var/lib/glance/images is where the images are stored and care should be taken.

3. Identity

/etc/keystone and /var/log/keystone follow the same rules as other components.

/var/lib/keystone, although it should not contain any data being used, can also be backed up just in case.

4. Block Storage

/etc/cinder and /var/log/cinder follow the same rules as other components.

/var/lib/cinder should also be backed up.

5. Networking

/etc/neutron and /var/log/neutron follow the same rules as other components.

/var/lib/neutron should also be backed up.

Cron Job for backup:

can easily automate this process by creating a cron job that runs the following script once per day:

For Controller

```
#!/bin/bash
backup_dir="/srv/node/sdc/backup-openstack/"
log_dir="/srv/node/sdc/backup-openstack/log"
lib_dir="/srv/node/sdc/backup-openstack/lib"
conf_dir="/srv/node/sdc/backup-openstack/conf"
filename="${backup_dir}/mysql-`hostname`-`eval date +%Y%m%d`.sql.gz"
# Dump the entire MySQL database
/usr/bin/mysqldump --opt --all-databases | gzip > $filename
# copy the all the directory to create the backup
cp -r /etc/chrony $conf_dir
cp -r /etc/memcached.conf $conf_dir
cp -r /etc/mysql $conf_dir
cp -r /etc/default/etcd $conf_dir
cp -r /etc/nova $conf_dir
cp -r /etc/keystone $conf_dir
cp -r /etc/glance $conf_dir
cp -r /etc/placement $conf_dir
cp -r /etc/neutron $conf_dir
cp -r /etc/cinder $conf_dir
cp -r /etc/swift $conf_dir
cp -r /etc/openstack-dashboard $conf_dir
#backup the lib directory files
cp -r /var/lib/chrony $lib_dir
cp -r /var/lib/nova $lib_dir
cp -r /var/lib/keystone $lib_dir
cp -r /var/lib/glance $lib_dir
cp -r /var/lib/placement $lib_dir
cp -r /var/lib/neutron $lib_dir
cp -r /var/lib/cinder $lib_dir
cp -r /var/lib/swift $lib_dir
#backup the lib directory files
cp -r /var/log/chrony $log_dir
cp -r /var/log/mysql $log_dir
cp -r /var/log/rabbitmq $log_dir
cp -r /var/log/nova $log_dir
cp -r /var/log/keystone $log_dir
cp -r /var/log/glance $log_dir
cp -r /var/log/placement $log_dir
cp -r /var/log/neutron $log_dir
cp -r /var/log/openvswitch $log_dir
cp -r /var/log/cinder $log_dir
cp -r /var/log/swift $log_dir
# Delete backups older than 7 days
find $backup_dir -ctime +3 -type f -delete
```

For Compute

```
#!/bin/bash
backup_dir="/part2/Openstack-backup/c3i-servercompute"
log_dir="/part2/Openstack-backup/c3i-servercompute/log"
lib_dir="/part2/Openstack-backup/c3i-servercompute/lib"
conf_dir="/part2/Openstack-backup/c3i-servercompute/conf"
pass="284961"

# create the backup of the configuration directory
sshpass -p "$pass" scp -r /etc/chrony k8@172.29.233.32:$conf_dir
sshpass -p "$pass" scp -r /etc/nova k8@172.29.233.32:$conf_dir
sshpass -p "$pass" scp -r /etc/neutron k8@172.29.233.32:$conf_dir
sshpass -p "$pass" scp -r /etc/cinder k8@172.29.233.32:$conf_dir

#backup the lib directory files
sshpass -p "$pass" scp -r /var/lib/chrony k8@172.29.233.32:$lib_dir
#scp -r /var/lib/nova $lib_dir
sshpass -p "$pass" scp -r /var/lib/neutron k8@172.29.233.32:$lib_dir
sshpass -p "$pass" scp -r /var/lib/cinder k8@172.29.233.32:$lib_dir

#backup the log directory files
sshpass -p "$pass" scp -r /var/log/chrony k8@172.29.233.32:$log_dir
sshpass -p "$pass" scp -r /var/log/nova k8@172.29.233.32:$log_dir
sshpass -p "$pass" scp -r /var/log/neutron k8@172.29.233.32:$log_dir
sshpass -p "$pass" scp -r /var/log/openvswitch k8@172.29.233.32:$log_dir
sshpass -p "$pass" scp -r /var/log/cinder k8@172.29.233.32:$log_dir

# Delete backups older than 7 days
find $backup_dir -ctime +7 -type f -delete
```

```
root@c3ihub: /home/k8
root@c3ihub:/home/k8# crontab -l
00 18 * * * TZ="Asia/Kolkata" /home/k8/Openstack-Installation/backup-openstack.sh >> /home/k8/cronjob.log 2>&1
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
root@c3ihub:/home/k8#
```

```
root@c3ihub: /part2/Openstack-backup/c3ihub-servercontroller
root@c3ihub:/home/k8# cd /part2
root@c3ihub:/part2# cd Openstack-backup/
root@c3ihub:/part2/Openstack-backup# ls
c3ihub-servercontroller  c3ihub-servercontroller
root@c3ihub:/part2/Openstack-backup# cd c3ihub-servercontroller/
root@c3ihub:/part2/Openstack-backup/c3ihub-servercontroller# ls
20241203  20241204  20241205  20241206  20241207  20241208  20241209  20241210
mysql-c3ihub-20241203.sql.gz  mysql-c3ihub-20241204.sql.gz  mysql-c3ihub-20241205.sql.gz  mysql-c3ihub-20241206.sql.gz  mysql-c3ihub-20241207.sql.gz  mysql-c3ihub-20241208.sql.gz  mysql-c3ihub-20241209.sql.gz  mysql-c3ihub-20241210.sql.gz
root@c3ihub:/part2/Openstack-backup/c3ihub-servercontroller#
```

Recovering Backups

1. To begin, first ensure that the service you are recovering is not running.

For example:

```
sudo service nova-api stop
sudo service nova-scheduler stop
sudo service nova-conductor stop
sudo service nova-novncproxy stop
sudo service nova-compute stop
```

2. Now, can import a previously backed-up database:

```
create nova;
create nova_api;
create nova_cell0;mysql -u root -p nova < path/to/backupfile.sql
use nova;
show tables;
```

3. can also restore backed-up nova directories

```
cp -a /path/to/backup/nova /etc/
```

4. Once the files are restored, start everything back up:

```
sudo service mysql restart  
sudo service nova-api restart  
sudo service nova-scheduler restart  
sudo service nova-conductor restart  
sudo service nova-novncproxy restart  
sudo service nova-compute restart
```

References

- <https://docs.openstack.org/install-guide/environment.html>
- <https://docs.openstack.org/nova/2024.2/install/>
- <https://docs.openstack.org/keystone/2024.2/install/>
- <https://docs.openstack.org/glance/2024.2/install/>
- <https://docs.openstack.org/cinder/2024.2/install/>
- <https://docs.openstack.org/horizon/2024.2/install/>
- <https://docs.openstack.org/neutron/2024.2/install/>
- <https://docs.openstack.org/placement/2024.2/install/>