OpenStack-Ansible Documentation: os_octavia role

Release 18.1.0.dev407

OpenStack-Ansible Contributors

Apr 14, 2026

CONTENTS

CHAPTER

ONE

CONFIGURING THE OCTAVIA LOAD BALANCING SERVICE

Octavia is an OpenStack project which provides operator-grade Load Balancing (as opposed to the namespace driver) by deploying each individual load balancer to its own instance and leveraging HAProxy to perform the load balancing.

Octavia is scalable and has built-in high availability through active-passive.

1.1 OpenStack-Ansible deployment

1. Create `br-lbaas` bridge on the controllers. Creating br-lbaas is done during the deployers host preparation and is out of scope of OpenStack-Ansible. Some explanation of how br-lbaas is used is given below.
2. Create the OpenStack-Ansible container(s) for Octavia. To do that you need to define hosts for `octavia-infra_hosts` group in `openstack_user_config.yml`. Once you do this, run the following playbook:

```
# openstack-ansible openstack.osa.containers_lxc_create --limit octavia_
↪all,octavia-infra_hosts
```

3. Define required overrides of the variables in defaults/main.yml of the OpenStack-Ansible Octavia role.
4. Run the OpenStack-Ansible Octavia playbook:

```
# openstack-ansible openstack.osa.octavia
```

5. Run the HAProxy playbook to add the new Octavia API endpoints to the load balancer.

```
# openstack-ansible openstack.osa.haproxy --tags haproxy-service-config
```

6. In order to enable Octavia dashboard panel in your Horizon dashboard, run Horizon playbook:

```
# openstack-ansible openstack.osa.horizon
```

1.2 Define project quota for Amphora driver

Amphora driver for Octavia is a default option on spawning Load Balancers with Octavia. The driver relies on OpenStack Nova/Neutron services to spawn instances from a specialized image which serve as Load Balancers.

These instances are created in a `service` project by default, and projects have no direct access to them.

With that operator must ensure, that the `service` project has sufficient quotas defined to handle all projects Load Balancers in it.

The suggested way of doing that is through leveraging the `openstack.osa.openstack_resources` playbook and defining following variables in `user_variables.yml` or `group_vars/utility_all`:

```
# In case of `octavia_loadbalancer_topology` set to ACTIVE_STANDBY (default)
# each Load Balancer will create 2 instances
_max_amphora_instances: 10000
openstack_user_identity:
   quotas:
     - name: "service"
       # Default Amphora flavor is 1 Core, 1024MB RAM
       cores: "{{ _max_amphora_instances }}"
       ram: "{{ (_max_amphora_instances | int) * 1024 }}"
       instances: "{{ _max_amphora_instances }}"
       port: "{{ (_max_amphora_instances | int) * 10 }}"
       server_groups: "{{ ((_max_amphora_instances | int) * 0.5) | int | abs }
↪}"
       server_group_members: 50
       # A security group is created per Load Balancer listener
       security_group: "{{ (_max_amphora_instances | int) * 1.5 | int | abs }}
↪"
       security_group_rule: "{{ ((_max_amphora_instances | int) * 1.5 | int |␣
↪abs) * 100 }}"
       # If `octavia_cinder_enabled: true` also define these
       volumes: "{{ _max_amphora_instances }}"
       # Volume size is defined with `octavia_cinder_volume_size` with default␣
↪of 20
       gigabytes: "{{ (_max_amphora_instances | int) * 20 }}"
```

These values will be applied on running `openstack-ansible openstack.osa.openstack_resources`, or as part of `openstack.osa.setup_openstack` playbook.

Warning

Octavia uses `ACTIVE_STANDBY` topology by default (`octavia_loadbalancer_topology`) and enables amphora anti-affinity (`octavia_enable_anti_affinity: true`). Deployments with only one compute host will fail to create load balancers.

1.3 Setup a neutron network for use by Octavia

Octavia needs connectivity between the control plane and the load balancing instances. For this purpose a provider network should be created which gives L2 connectivity between the octavia services on the controllers (either containerised or deployed on metal) and the Octavia amphora instances. Refer to the appropriate documentation for the octavia service and consult the tests in this project for a working example.

Special attention needs to be applied to the provider network `--allocation-pool` not to have ip addresses which overlap with those assigned to hosts, lxc containers or other infrastructure such as routers or firewalls which may be in use.

An example which gives `172.29.232.0-9/22` to the OSA dynamic inventory and the remainder of the addresses to the neutron allocation pool without overlap is as follows:

In `openstack_user_config.yml` the following:

```
#the address range for the whole lbaas network
cidr_networks:
   lbaas: 172.29.232.0/22

#the range of ip addresses excluded from the dynamic inventory
used_ips:
   - "172.29.232.10,172.29.235.200"
```

And define in `user_variables.yml`:

```
#the range of addresses which neutron can allocate for amphora instances
octavia_management_net_subnet_allocation_pools: "172.29.232.10-172.29.235.200"
```

Note

The system will deploy an iptables firewall if `octavia_ip_tables_fw` is set to `true` (the default). This adds additional protection to the control plane in the rare instance a load balancing vm is compromised. Please review carefully the rules and adjust them for your installation. Please be aware that logging of dropped packages is not enabled and you will need to add those rules manually.

1.3.1 FLAT networking scenario

In a general case, neutron networking can be a simple flat network. However in a complex case, this can be whatever you need and want. Ensure you adjust the deployment accordingly. An example entry into `openstack_user_config.yml` is shown below:

```
- network:
   container_bridge: "br-lbaas"
   container_type: "veth"
   container_interface: "eth14"
   host_bind_override: "bond0"  # Defines neutron physical network mapping
   ip_from_q: "octavia"
   type: "flat"
   net_name: "octavia"
   group_binds:
     - octavia_worker
     - octavia_housekeeping
     - octavia_health_manager
     # in case of OVN
     - neutron_ovn_gateway
     # in case of OVS
     - neutron_openvswitch_agent
```

There are a couple of variables which need to be adjusted if you dont use `lbaas` for the provider network name and `lbaas-mgmt` for the neutron name. Furthermore, the system tries to infer certain values based on the inventory which might not always work and hence might need to be explicitly declared. Review the file `defaults/main.yml` for more information.

The Octavia Ansible role can create the required neutron networks itself. Please review the corresponding settings - especially `octavia_management_net_subnet_cidr` should be adjusted to suit your environment. Alternatively, the neutron network can be pre-created elsewhere and consumed by Octavia.

1.3.2 VLAN networking scenario

In case you want to leverage standard vlan networking for the Octavia management network the definition in `openstack_user_config.yml` may look like this:

```
- network:
    container_bridge: "br-lbaas"
    container_type: "veth"
    container_interface: "eth14"
    ip_from_q: "lbaas"
    type: "raw"
    net_name: lbaas
    group_binds:
      - octavia_worker
      - octavia_housekeeping
      - octavia_health_manager
      # in case of OVN
      - neutron_ovn_gateway
      # in case of OVS
      - neutron_openvswitch_agent
```

Add extend `user_variables.yml` with following overrides:

```
octavia_provider_network_name: vlan
octavia_provider_network_type: vlan
octavia_provider_segmentation_id: 400
octavia_provider_inventory_net_name: lbaas
```

In addition to this, you will need to ensure that you have an interface that links neutron-managed br-vlan with br-lbaas on the controller nodes (for the case when br-vlan already exists on the controllers when they also host the neutron L3 agent). Making veth pairs or macvlans for this might be suitable.

1.4 Building Octavia images

Note

The default behavior is to download a test image from the OpenStack artifact storage the Octavia team provides daily. Because this image doesnt apply operating system security patches in a timely manner it is unsuited for production use.

Some Operating System vendors might provide official amphora builds or an organization might maintain their own artifact storage - for those cases the automatic download can be leveraged, too.

Images using the `diskimage-builder` must be built outside of a container. For this process, use one of the physical hosts within the environment.

1. Install the necessary packages and configure a Python virtual environment

```
apt-get install qemu-system uuid-runtime curl kpartx git jq python3-pip
pip3 install virtualenv

virtualenv -p /usr/bin/python3 /opt/octavia-image-build
source /opt/octavia-image-build/bin/activate
```

2. Clone the necessary repositories and dependencies

```
# git clone https://opendev.org/openstack/octavia

  /opt/octavia-image-build/bin/pip install --isolated \
  git+https://opendev.org/openstack/diskimage-builder
```

3. Run Octavias diskimage script

   In the `octavia/diskimage-create` directory run:

```
./diskimage-create.sh
```

   Disable `octavia-image-build` venv:

```
deactivate
```

4. Upload the created user images into the Image (glance) Service:

```
openstack image create --disk-format qcow2 \
   --container-format bare --tag octavia-amphora-image --file amphora-x64-
↪haproxy.qcow2 \
   --private --project service amphora-x64-haproxy
```

> **Note**
>
> Alternatively you can specify the new image in the appropriate settings and rerun the ansible with an appropriate tag.

You can find more information about the diskimage script and the process at https://opendev.org/openstack/octavia/src/branch/master/diskimage-create

Here is a script to perform all those tasks at once:

```
#!/usr/bin/env bash
set -euo pipefail

VENV_DIR="/opt/octavia-image-build"
WORK_DIR="/tmp"
OCTAVIA_REPO="${WORK_DIR}/octavia"
OUTPUT_IMAGE="amphora-x64-haproxy.qcow2"

echo "Installing required packages..."
apt-get update
apt-get install -y \
```

(continues on next page)

(continued from previous page)

```
    qemu-system \
    uuid-runtime \
    curl \
    kpartx \
    git \
    jq \
    python3-pip \
    python3-virtualenv \
    debootstrap

echo "Creating Python virtual environment..."
virtualenv -p /usr/bin/python3 "${VENV_DIR}"

source "${VENV_DIR}/bin/activate"

echo "Cloning Octavia repository..."
cd "${WORK_DIR}"
rm -rf "${OCTAVIA_REPO}"
git clone https://opendev.org/openstack/octavia "${OCTAVIA_REPO}"

echo "Installing diskimage-builder..."
"${VENV_DIR}/bin/pip" install --isolated \
    git+https://opendev.org/openstack/diskimage-builder

echo "Building Amphora image..."
cd "${OCTAVIA_REPO}/diskimage-create"
./diskimage-create.sh

echo "Moving output image to ${WORK_DIR}..."
mv "${OUTPUT_IMAGE}" "${WORK_DIR}/"

deactivate

echo "Done. Image available at: ${WORK_DIR}/${OUTPUT_IMAGE}"
```

And upload image:

```
openstack image delete amphora-x64-haproxy
openstack image create --disk-format qcow2 \
  --container-format bare --tag octavia-amphora-image --file /tmp/
↪amphora-x64-haproxy.qcow2 \
  --private --project service amphora-x64-haproxy
```

1.5 Creating the cryptographic certificates

Note

For production installation make sure that you review this very carefully with your own security

requirements and potentially use your own CA to sign the certificates.

The system will automatically generate and use self-signed certificates with different Certificate Authorities for control plane and amphora. Make sure to store a copy in a safe place for potential disaster recovery.

1.6 Optional: Configuring Octavia with ssh access to the amphora

In rare cases it might be beneficial to gain ssh access to the amphora for additional trouble shooting. Follow these steps to enable access.

1. Configure Octavia accordingly

   Add a `octavia_ssh_enabled: True` to the user file in /etc/openstack-deploy

2. Run `os_octavia` role. SSH key will be generated and uploaded

Note

SSH key will be stored on the `octavia_keypair_setup_host` (which by default is `localhost`) in `~/.ssh/{{ octavia_ssh_key_name }}`

1.7 Optional: Tuning Octavia for production use

We suggest setting more specific `octavia_cert_dir` to prevent accidental certificate rotation.

CHAPTER

TWO

BASIC OCTAVIA LOAD BALANCER TROUBLESHOOTING GUIDE

This guide provides a quick and practical walkthrough for creating and troubleshooting a basic HTTP load balancer using OpenStack Octavia.

2.1 Scenario Overview

We assume:

1. Two backend instances exist in subnet `private-subnet-1 (10.0.2.0/24)`
2. Instances are reachable on port 80 (e.g., simple web servers running)
3. The subnet is connected to a router (required for Floating IP access)

2.2 Creating Load Balancer

1. Create the load balancer:

```
# openstack loadbalancer create --name load-balancer-1 --vip-subnet-id␣
↪private-subnet-1
```

2. Create listener:

```
# openstack loadbalancer listener create \
    --name listener-1 \
    --protocol HTTP \
    --protocol-port 80 \
    load-balancer-1
```

3. Create pool (using Round Robin algorithm):

```
# openstack loadbalancer pool create \
    --name pool-1 \
    --lb-algorithm ROUND_ROBIN \
    --listener listener-1 \
    --protocol HTTP
```

4. Add members:

```
# openstack loadbalancer member create \
    --subnet-id private-subnet-1 \
```

(continues on next page)

(continued from previous page)

```
    --address 10.0.2.252 \
    --protocol-port 80 \
    pool-1

# openstack loadbalancer member create \
    --subnet-id private-subnet-1 \
    --address 10.0.2.57 \
    --protocol-port 80 \
    pool-1
```

2.3 Assign Floating IP to VIP

Ensure your subnet is connected to a router, then assign a Floating IP:

```
# openstack floating ip set --port <load_balancer_vip_port_id> <floating_ip_
↪id>
```

2.4 Verification in Horizon / CLI

After successful creation, two Amphora instances will be created (ACTIVE_STANDBY setup):

- MASTER
- BACKUP

You can view them via CLI:

```
# openstack loadbalancer amphora list
```

Typical output looks like:

```
+--------------------------------------+--------------------------------------
↪+-----------+--------+---------------+-----------+
| id                                   | loadbalancer_id                      ␣
↪| status    | role   | lb_network_ip  | ha_ip     |
+--------------------------------------+--------------------------------------
↪+-----------+--------+---------------+-----------+
| 3b42cd08-564f-4415-b7cb-9c6e3920f2d4 | c66e66c9-9904-4a07-bc88-3eaa24d2c3ef␣
↪| ALLOCATED | MASTER | 172.29.232.161 | 10.0.2.64 |
| 62172a88-08d4-4a12-8a9a-4721d66e1544 | c66e66c9-9904-4a07-bc88-3eaa24d2c3ef␣
↪| ALLOCATED | BACKUP | 172.29.233.104 | 10.0.2.64 |
+--------------------------------------+--------------------------------------
↪+-----------+--------+---------------+-----------+
```

- `172.29.232.0/22` -> management network (lb-mgmt-net)
- `10.0.2.64` -> VIP address

2.5 Connecting to Amphora (Master)

Ensure:

- SSH access is enabled
- You have connectivity to the management network

```
# ssh -i .ssh/octavia_key ubuntu@172.29.232.161
```

2.6 Inspect Network Configuration

Default namespace (management interface):

```
ubuntu@amphora:~$ ip a
```

Sample output:

```
ens3: <BROADCAST,MULTICAST,UP,LOWER_UP>
    inet 172.29.232.161/22
```

This interface belongs to the **management network**.

2.7 Network Namespaces

Octavia uses a dedicated namespace for HAProxy:

```
ubuntu@amphora:~$ sudo ip netns list
amphora-haproxy (id: 0)
```

Inspect interfaces inside the namespace:

```
ubuntu@amphora:~$ sudo ip netns exec amphora-haproxy ip a
```

Sample output:

```
eth1: <BROADCAST,MULTICAST,UP,LOWER_UP>
      inet 10.0.2.69/24
      inet 10.0.2.64/32
```

- `10.0.2.64` -> VIP address (active on MASTER)
- `10.0.2.69` -> Amphora instance IP

The VIP is configured as a /32 and moves between MASTER/BACKUP during failover.

2.8 HAProxy Service Inspection

Each apmhora instance runs its own HAProxy service.

1. Check service status:

   ```
   ubuntu@amphora:~$ sudo systemctl status haproxy-<load-balancer-id>
   ```

2. Check configuration:

```
ubuntu@amphora:~$ sudo cat /var/lib/octavia/<load-balancer-id>/haproxy.cfg
```

Configuration snippet:

```
peers c66e66c999044a07bc883eaa24d2c3ef_peers
    peer cbZyp_Dj-MpsKs2NJeavavY7dbU 10.0.2.69:1025
    peer 9PCRPduKu_p18PXO49ZHWE1bApI 10.0.2.200:1025

frontend 265a327c-4486-484a-a471-8fdd597c1911
    bind 10.0.2.64:80
    mode http
    default_backend a972f1f9-9f9f-4b48-af21-276f3e0d22f3:265a327c-4486-
↪484a-a471-8fdd597c1911

backend a972f1f9-9f9f-4b48-af21-276f3e0d22f3:265a327c-4486-484a-a471-
↪8fdd597c1911
    balance roundrobin
    server efa07206-9039-42fe-87a8-6dd57818b1e2 10.0.2.252:80 weight 1
    server 74c9b591-b816-4cf7-8e87-efa76ab5bcaf 10.0.2.57:80 weight 1
```

Key lines:

- `peers ...` -> defines synchronization between Amphora nodes for HA state sharing
- `peer <id> <ip>:1025` -> remote Amphora nodes participating in sync
- `bind 10.0.2.64:80` -> HAProxy listens on VIP address
- `mode http` -> layer 7 load balancing
- `default_backend ...` -> directs traffic to backend pool
- `balance roundrobin` -> distributes requests evenly across members
- `server <id> <ip>:80` -> backend instances receiving traffic
- `weight 1` -> equal traffic distribution

2.9 Common Troubleshooting Checks

2.9.1 1. VIP is not reachable

- Check Floating IP assignment
- Verify router connectivity
- Confirm security groups allow port 80

2.9.2 2. No traffic to backend

- Verify backend instances are running web servers
- Test directly:

```
ubuntu@amphora:~$ curl -I http://10.0.2.252
ubuntu@amphora:~$ curl -I http://10.0.2.57
```

2.9.3 3. HAProxy not running

- Check service:

```
ubuntu@amphora:~$ sudo systemctl status haproxy-<load-balancer-id>
```

2.9.4 4. Wrong IP binding

- Ensure VIP exists in namespace:

```
ubuntu@amphora:~$ sudo ip netns exec amphora-haproxy ip a
```

This is a **basic Octavia troubleshooting workflow** which serves as a starting point for debugging load balancer issues in OpenStack environments.

This is an OpenStack-Ansible role to deploy the Octavia Load Balancing service.

To clone or view the source code for this repository, visit the role repository for os_octavia.

CHAPTER

THREE

DEFAULT VARIABLES

```yaml
## Verbosity Options
debug: false

# Set the host which will execute the shade modules
# for the service setup. The host must already have
# clouds.yaml properly configured.
octavia_service_setup_host: "{{ openstack_service_setup_host | default(
↪'localhost') }}"
octavia_service_setup_host_python_interpreter: >-
  {{
    openstack_service_setup_host_python_interpreter | default(
      (octavia_service_setup_host == 'localhost') | ternary(ansible_playbook_
↪python, ansible_facts['python']['executable']))
  }}

# Set installation method.
octavia_install_method: "{{ service_install_method | default('source') }}"
octavia_venv_python_executable: "{{ openstack_venv_python_executable |␣
↪default('python3') }}"

## Allow TLS listener
octavia_tls_listener_enabled: true

# Set the package install state for distribution packages
# Options are 'present' and 'latest'
octavia_package_state: "{{ package_state | default('latest') }}"

# Source git repo/branch settings
octavia_git_repo: https://opendev.org/openstack/octavia
octavia_git_install_branch: master
octavia_upper_constraints_url: >-
  {{ requirements_git_url | default('https://releases.openstack.org/
↪constraints/upper/' ~ requirements_git_install_branch | default('master')) }
↪}

octavia_ovn_octavia_provider_git_repo: https://opendev.org/openstack/ovn-
↪octavia-provider
octavia_ovn_octavia_provider_git_install_branch: master
```

(continues on next page)

(continued from previous page)

```
octavia_git_constraints:
  - "--constraint {{ octavia_upper_constraints_url }}"

octavia_pip_install_args: "{{ pip_install_options | default('') }}"

# Name of the virtual env to deploy into
octavia_venv_tag: "{{ venv_tag | default('untagged') }}"
octavia_bin: "{{ _octavia_bin }}"

octavia_clients_endpoint: internal

octavia_auth_strategy: keystone

## Barbican certificates
octavia_barbican_enabled: false

## Cinder Volume
octavia_cinder_enabled: false
cinder_default_availability_zone: "{{ octavia_amp_availability_zone }}"
octavia_cinder_volume_size: 20
octavia_cinder_volume_type: "volumes-hdd"

## Database info
octavia_db_setup_host: "{{ openstack_db_setup_host | default('localhost') }}"
octavia_db_setup_python_interpreter: >-
  {{
    openstack_db_setup_python_interpreter | default(
      (octavia_db_setup_host == 'localhost') | ternary(ansible_playbook_
↪python, ansible_facts['python']['executable']))
  }}
octavia_galera_address: "{{ galera_address | default('127.0.0.1') }}"
octavia_galera_user: octavia
octavia_galera_database: octavia
octavia_galera_persistence_database: octavia_persistence
octavia_galera_use_ssl: "{{ galera_use_ssl | default(False) }}"
octavia_galera_ssl_ca_cert: "{{ galera_ssl_ca_cert | default('') }}"
octavia_db_max_overflow: "{{ openstack_db_max_overflow | default('50') }}"
octavia_db_max_pool_size: "{{ openstack_db_max_pool_size | default('5') }}"
octavia_db_pool_timeout: "{{ openstack_db_pool_timeout | default('30') }}"
octavia_db_connection_recycle_time: "{{ openstack_db_connection_recycle_time␣
↪| default('600') }}"
octavia_galera_port: "{{ galera_port | default('3306') }}"

## Coordination info
# NOTE: Only Zookeeper and Redis are supported for Octavia
octavia_coordination_driver: "{{ coordination_driver | default('zookeeper') }}
↪"
octavia_coordination_group: "{{ coordination_host_group | default('zookeeper_
```

(continues on next page)

(continued from previous page)

```
↪all') }}"
# octavia_coordination_enable: "{{ octavia_coordination_group in groups and␣
↪groups[octavia_coordination_group] | length > 0 }}"
octavia_coordination_enable: false  # Temporarily disabled due to ZooKeeper/
↪Kazoo instability; re-enable after issue is resolved
octavia_coordination_namespace: octavia_jobboard
octavia_coordination_client_ssl: "{{ coordination_client_ssl | default(False)␣
↪}}"
octavia_coordination_verify_cert: "{{ coordination_verify_cert |␣
↪default(True) }}"
octavia_coordination_port: "{{ coordination_port | default(octavia_
↪coordination_client_ssl | ternary('2281', '2181')) }}"

## Oslo Messaging

# RPC
octavia_oslomsg_rpc_host_group: "{{ oslomsg_rpc_host_group | default(
↪'rabbitmq_all') }}"
octavia_oslomsg_rpc_setup_host: "{{ (octavia_oslomsg_rpc_host_group in␣
↪groups) | ternary(groups[octavia_oslomsg_rpc_host_group][0], 'localhost') }}
↪"
octavia_oslomsg_rpc_transport: "{{ oslomsg_rpc_transport | default('rabbit') }
↪}"
octavia_oslomsg_rpc_servers: "{{ oslomsg_rpc_servers | default('127.0.0.1') }}
↪"
octavia_oslomsg_rpc_port: "{{ oslomsg_rpc_port | default('5672') }}"
octavia_oslomsg_rpc_use_ssl: "{{ oslomsg_rpc_use_ssl | default(False) }}"
octavia_oslomsg_rpc_userid: octavia
octavia_oslomsg_rpc_policies: []
# vhost name depends on value of oslomsg_rabbit_quorum_queues. In case quorum␣
↪queues
# are not used - vhost name will be prefixed with leading `/`.
octavia_oslomsg_rpc_vhost:
  - name: /octavia
    state: "{{ octavia_oslomsg_rabbit_quorum_queues | ternary('absent',
↪'present') }}"
  - name: octavia
    state: "{{ octavia_oslomsg_rabbit_quorum_queues | ternary('present',
↪'absent') }}"
octavia_oslomsg_rpc_ssl_version: "{{ oslomsg_rpc_ssl_version | default('TLSv1_
↪2') }}"
octavia_oslomsg_rpc_ssl_ca_file: "{{ oslomsg_rpc_ssl_ca_file | default('') }}"

# Notify
octavia_oslomsg_notify_configure: "{{ oslomsg_notify_configure |␣
↪default(octavia_ceilometer_enabled) }}"
octavia_oslomsg_notify_host_group: "{{ oslomsg_notify_host_group | default(
↪'rabbitmq_all') }}"
octavia_oslomsg_notify_setup_host: >-
```

(continues on next page)

(continued from previous page)

```
  {{ (octavia_oslomsg_notify_host_group in groups) | ternary(groups[octavia_
↪oslomsg_notify_host_group][0], 'localhost') }}
octavia_oslomsg_notify_transport: "{{ oslomsg_notify_transport | default(
↪'rabbit') }}"
octavia_oslomsg_notify_servers: "{{ oslomsg_notify_servers | default('127.0.0.
↪1') }}"
octavia_oslomsg_notify_port: "{{ oslomsg_notify_port | default('5672') }}"
octavia_oslomsg_notify_use_ssl: "{{ oslomsg_notify_use_ssl | default(False) }}
↪"
octavia_oslomsg_notify_userid: "{{ octavia_oslomsg_rpc_userid }}"
octavia_oslomsg_notify_password: "{{ octavia_oslomsg_rpc_password }}"
octavia_oslomsg_notify_vhost: "{{ octavia_oslomsg_rpc_vhost }}"
octavia_oslomsg_notify_ssl_version: "{{ oslomsg_notify_ssl_version | default(
↪'TLSv1_2') }}"
octavia_oslomsg_notify_ssl_ca_file: "{{ oslomsg_notify_ssl_ca_file | default('
↪') }}"
octavia_oslomsg_notify_policies: []

## RabbitMQ integration
octavia_oslomsg_rabbit_quorum_queues: "{{ oslomsg_rabbit_quorum_queues |␣
↪default(True) }}"
octavia_oslomsg_rabbit_stream_fanout: "{{ oslomsg_rabbit_stream_fanout |␣
↪default(octavia_oslomsg_rabbit_quorum_queues) }}"
octavia_oslomsg_rabbit_transient_quorum_queues: "{{ oslomsg_rabbit_transient_
↪quorum_queues | default(octavia_oslomsg_rabbit_stream_fanout) }}"
octavia_oslomsg_rabbit_qos_prefetch_count: "{{ oslomsg_rabbit_qos_prefetch_
↪count | default(octavia_oslomsg_rabbit_stream_fanout | ternary(10, 0)) }}"
octavia_oslomsg_rabbit_queue_manager: "{{ oslomsg_rabbit_queue_manager |␣
↪default(octavia_oslomsg_rabbit_quorum_queues) }}"
octavia_oslomsg_rabbit_quorum_delivery_limit: "{{ oslomsg_rabbit_quorum_
↪delivery_limit | default(0) }}"
octavia_oslomsg_rabbit_quorum_max_memory_bytes: "{{ oslomsg_rabbit_quorum_max_
↪memory_bytes | default(0) }}"

octavia_ceilometer_enabled: "{{ (groups['ceilometer_all'] is defined) and␣
↪(groups['ceilometer_all'] | length > 0) }}"

## octavia User / Group
octavia_system_user_name: octavia
octavia_system_group_name: octavia
octavia_system_shell: /bin/false
octavia_system_comment: octavia system user
octavia_system_home_folder: "/var/lib/{{ octavia_system_user_name }}"
octavia_system_slice_name: octavia
octavia_lock_dir: "{{ openstack_lock_dir | default('/run/lock') }}"

## Auth
octavia_service_region: "{{ service_region | default('RegionOne') }}"
octavia_service_project_name: "service"
```

(continues on next page)

(continued from previous page)

```
octavia_service_user_name: "octavia"
octavia_service_role_names:
  - admin
  - service
octavia_service_token_roles:
  - service
octavia_service_token_roles_required: "{{ openstack_service_token_roles_
↪required | default(True) }}"
octavia_service_project_domain_id: default
octavia_service_user_domain_id: default
octavia_keystone_auth_plugin: "{{ octavia_keystone_auth_type }}"
octavia_keystone_auth_type: password

## octavia api service type and data
octavia_service_name: octavia
octavia_service_description: "Octavia Load Balancing Service"
octavia_service_port: 9876
octavia_service_proto: http
octavia_service_publicuri_proto: "{{ openstack_service_publicuri_proto |␣
↪default(octavia_service_proto) }}"
octavia_service_adminuri_proto: "{{ openstack_service_adminuri_proto |␣
↪default(octavia_service_proto) }}"
octavia_service_internaluri_proto: "{{ openstack_service_internaluri_proto |␣
↪default(octavia_service_proto) }}"
octavia_service_type: load-balancer
octavia_service_publicuri: "{{ octavia_service_publicuri_proto }}://{{␣
↪external_lb_vip_address }}:{{ octavia_service_port }}"
octavia_service_adminuri: "{{ octavia_service_adminuri_proto }}://{{ internal_
↪lb_vip_address }}:{{ octavia_service_port }}"
octavia_service_internaluri: "{{ octavia_service_internaluri_proto }}://{{␣
↪internal_lb_vip_address }}:{{ octavia_service_port }}"

octavia_service_in_ldap: "{{ service_ldap_backend_enabled | default(False) }}"

## RPC
octavia_rpc_thread_pool_size: 64
octavia_rpc_conn_pool_size: 30

## Timeouts
octavia_amp_active_retries: 10

## Plugin dirs
octavia_plugin_dirs:
  - /usr/lib/octavia
  - /usr/local/lib/octavia

###
### Python code details
###
```

(continues on next page)

(continued from previous page)

```
octavia_pip_packages:
  - cryptography
  - keystonemiddleware
  - osprofiler
  - PyMySQL
  - pymemcache
  - python-glanceclient
  - python-keystoneclient
  - python-memcached
  - python-neutronclient
  - python-novaclient
  - python-openstackclient
  - python-octaviaclient
  - "git+{{ octavia_git_repo }}@{{ octavia_git_install_branch }}#egg=octavia"
  - systemd-python
  - "tooz[{{ octavia_coordination_driver }}]"

# Specific pip packages provided by the user
octavia_user_pip_packages: []

octavia_optional_ovn_octavia_provider_pip_packages:
  - "git+{{ octavia_ovn_octavia_provider_git_repo }}@{{ octavia_ovn_octavia_
↪provider_git_install_branch }}#egg=ovn-octavia-provider"

# Memcached override
octavia_memcached_servers: "{{ memcached_servers }}"

octavia_api_init_overrides: {}
octavia_worker_init_overrides: {}
octavia_housekeeping_init_overrides: {}
octavia_health_manager_init_overrides: {}
octavia_driver_agent_init_overrides:
  Service:
    Killmode: process

## Service Name-Group Mapping
octavia_services:
  octavia-api:
    group: octavia_api
    service_name: octavia-api
    start_order: 4
    init_config_overrides: "{{ octavia_api_init_overrides }}"
    wsgi_app: true
    wsgi: "octavia.wsgi.api:application"
    uwsgi_overrides: "{{ octavia_api_uwsgi_ini_overrides }}"
    uwsgi_port: "{{ octavia_service_port }}"
    uwsgi_bind_address: "{{ octavia_uwsgi_bind_address }}"
    uwsgi_tls: "{{ octavia_backend_ssl | ternary(octavia_uwsgi_tls, {}) }}"
```

(continues on next page)

(continued from previous page)

```
  octavia-worker:
    group: octavia_worker
    service_name: octavia-worker
    start_order: 1
    init_config_overrides: "{{ octavia_worker_init_overrides }}"
    execstarts: "{{ octavia_bin }}/octavia-worker"
    execreloads: "/bin/kill -HUP $MAINPID"
  octavia-housekeeping:
    group: octavia_housekeeping
    service_name: octavia-housekeeping
    start_order: 3
    init_config_overrides: "{{ octavia_housekeeping_init_overrides }}"
    execstarts: "{{ octavia_bin }}/octavia-housekeeping"
    execreloads: "/bin/kill -HUP $MAINPID"
  octavia-health-manager:
    group: octavia_health_manager
    service_name: octavia-health-manager
    start_order: 2
    init_config_overrides: "{{ octavia_health_manager_init_overrides }}"
    execstarts: "{{ octavia_bin }}/octavia-health-manager"
    execreloads: "/bin/kill -HUP $MAINPID"
  octavia-driver-agent:
    group: octavia_api
    service_name: octavia-driver-agent
    service_en: "{{ octavia_ovn_enabled }}"
    start_order: 5
    init_config_overrides: "{{ octavia_driver_agent_init_overrides }}"
    execstarts: "{{ octavia_bin }}/octavia-driver-agent --config-file /etc/
↪octavia/octavia.conf"
    execreloads: "/bin/kill -HUP $MAINPID"

# Required secrets for the role
octavia_required_secrets:
  - keystone_auth_admin_password
  - octavia_container_mysql_password
  - octavia_oslomsg_rpc_password
  - octavia_oslomsg_notify_password
  - octavia_service_password
  - memcached_encryption_key

## Octavia configs
# Load balancer topology options are SINGLE, ACTIVE_STANDBY
octavia_loadbalancer_topology: ACTIVE_STANDBY

# Image tag for the amphora image in glance
octavia_glance_image_tag: octavia-amphora-image
# add here the id of the image owner to avoid faked images being used
octavia_amp_image_owner_id:
# download the image from an artefact server
```

(continues on next page)

(continued from previous page)

```
# Note: The default is the Octavia test image so don't use that in prod
octavia_download_artefact: true
# The URL to download from
octavia_artefact_url: http://tarballs.openstack.org/octavia/test-images/test-
↪only-amphora-x64-haproxy-ubuntu-noble.qcow2
# Set the directory where the downloaded image will be stored
# on the octavia_service_setup_host host. If the host is localhost,
# then the user running the playbook must have access to it.
octavia_amp_image_path: "{{ lookup('env', 'HOME', default='/root') }}/
↪openstack-ansible/octavia"
octavia_amp_image_path_owner: "{{ lookup('env', 'USER', default='root') }}"
# enable uploading image to glance automatically
octavia_amp_image_upload_enabled: "{{ octavia_download_artefact }}"
octavia_amp_image_resource:
  - name: amphora-x64-haproxy
    url: "{{ octavia_artefact_url }}"
    # Image checksum is required for rotating old images
    # checksum:
    disk_format: qcow2
    keep_copies: 1
    tags:
      - "{{ octavia_glance_image_tag }}"
    owner: "{{ octavia_service_project_name }}"
    owner_domain: "{{ octavia_service_project_domain_id }}"
    image_download_path: "{{ octavia_amp_image_path }}"

# Name of the Octavia security group
octavia_security_group_name: octavia_sec_grp
# Additional rules to add to the security group for the amphora
octavia_security_group_additional_rules: []
# Restrict access to only authorized hosts
octavia_security_group_rule_cidr: "{{ octavia_management_net_subnet_cidr }}"

octavia_resources_deploy_host: localhost
octavia_resources_deploy_python_interpreter: "{{ ansible_playbook_python }}"
# ssh enabled - switch to True if you need ssh access to the amphora
octavia_ssh_enabled: false
octavia_ssh_key_manage: true
octavia_ssh_key_name: octavia_key
octavia_ssh_key_dir: "{{ lookup('env', 'HOME', default='/root') ~ '/.ssh' }}"
# SSH Key variables below are set to "old" values for backwards compatability
# of how Nova used to generate keypairs.
octavia_ssh_key_comment: Generated-by-Nova
# Options: ssh, pkcs1 and pkcs8
octavia_ssh_key_format: ssh
# Options: rsa, dsa, rsa1, ecdsa, ed25519
octavia_ssh_key_type: rsa
octavia_ssh_key_size: 2048
# port the agent listens on
```

(continues on next page)

(continued from previous page)

```
octavia_agent_port: "9443"
octavia_health_manager_port: 5555

# Octavia Nova flavor
octavia_amp_flavor_name: "m1.amphora"
octavia_amp_ram: 1024
octavia_amp_vcpu: 1
octavia_amp_disk: "{{ octavia_cinder_enabled | ternary(0, 20) }}"
# octavia_amp_extra_specs:

# only increase when it's a really busy system since this is by deployed host,
# e.g. 3 hosts, 5 workers (this param) per host, results in 15 worker total
octavia_task_flow_max_workers: 5

# Enable provisioning status sync with neutron db
octavia_sync_provisioning_status: false

# this controls if Octavia should add an anti-affinity hint to make sure
# two amphora are not placed pn the same host (the most common setup of
# ant affinity features in Nova).
octavia_enable_anti_affinity: true

# Some installations put hardware more suited for load balancing in special
# availability zones. This allows to target a specific availability zone
# for amphora creation
octavia_amp_availability_zone: nova

# List of haproxy template files to copy from deployment host to octavia hosts
# octavia_user_haproxy_templates:
#   - src: "/etc/openstack_deploy/octavia/haproxy_templates/base.cfg.j2"
#     dest: "/etc/octavia/templates/base.cfg.j2"
#   - src: "/etc/openstack_deploy/octavia/haproxy_templates/haproxy.cfg.j2"
#     dest: "/etc/octavia/templates/haproxy.cfg.j2"
#   - src: "/etc/openstack_deploy/octavia/haproxy_templates/macros.cfg.j2"
#     dest: "/etc/octavia/templates/macros.cfg.j2"
octavia_user_haproxy_templates: []
# Path of custom haproxy template file
# octavia_haproxy_amphora_template: /etc/octavia/templates/haproxy.cfg.j2

# Name of the Octavia management network in Neutron
octavia_neutron_management_network_name: lbaas-mgmt
# Name of the Neutron provider net in the system (flat, vlan, ...)
octavia_provider_network_name: lbaas
# Network type
octavia_provider_network_type: flat
# Network segmentation ID if vlan, gre...
# octavia_provider_segmentation_id:
# Network CIDR
octavia_management_net_subnet_cidr: 172.29.232.0/22
```

(continues on next page)

(continued from previous page)

```
# Example allocation range:
# octavia_management_net_subnet_allocation_pools: "172.29.232.10-172.29.235.
↪200"
octavia_management_net_subnet_allocation_pools: ""
# Do we require the Neutron DHCP server
octavia_management_net_dhcp: "False"
# Should Octavia set up the network and subnet?
octavia_service_net_setup: true
# This should match net_name from provider_networks structure in openstack_
↪user_config
octavia_provider_inventory_net_name: "{{ octavia_provider_network_name }}"
# This gets container managment network structure based on octavia_provider_
↪inventory_net_name
octavia_provider_network: >-
  {{ provider_networks | map(attribute='network') | selectattr('net_name',
↪'defined') | selectattr(
    'net_name', 'equalto', octavia_provider_inventory_net_name) | list | first
  }}
# The name of the network address pool
octavia_container_network_name: "{{ octavia_provider_network['ip_from_q'] }}_
↪address"
octavia_hm_group: "octavia_health_manager"
# Note: We use some heuristics here but if you do anything special make sure␣
↪to use the
# ip addresses on the right network. This will use the container networking␣
↪to figure out the ip
octavia_hm_hosts: >-
  {% for host in groups[octavia_hm_group] %}{{ hostvars[host]['container_
↪networks'][octavia_container_network_name]['address'] }}{%
    if not loop.last %},{% endif %}{% endfor %}
# Set this to the right container port aka the eth you connect to the octavia
# management network
octavia_container_interface: "{{ octavia_provider_network.container_interface␣
↪}}"
# Set this to true to drop the iptables rules
octavia_ip_tables_fw: true
# The iptable rules
octavia_iptables_rules:
  # Allow icmp
  - chain: INPUT
    protocol: icmp
    ctstate: NEW
    icmp_type: 8
    jump: ACCEPT
  # Allow existing connections:
  - chain: INPUT
    in_interface: "{{ octavia_container_interface }}"
    ctstate: RELATED,ESTABLISHED
    jump: ACCEPT
```

(continues on next page)

(continued from previous page)

```
  # Allow heartbeat:
  - chain: INPUT
    in_interface: "{{ octavia_container_interface }}"
    protocol: udp
    destination_port: "{{ octavia_health_manager_port }}"
    jump: ACCEPT
  # Reject INPUT:
  - chain: INPUT
    in_interface: "{{ octavia_container_interface }}"
    reject_with: icmp-port-unreachable
  # Reject FORWARD:
  - chain: FORWARD
    in_interface: "{{ octavia_container_interface }}"
    reject_with: icmp-port-unreachable
  # Allow icmp6
  - chain: INPUT
    protocol: icmpv6
    jump: ACCEPT
    ip_version: ipv6
  # Allow existing connections
  - chain: INPUT
    in_interface: "{{ octavia_container_interface }}"
    ctstate: RELATED,ESTABLISHED
    jump: ACCEPT
    ip_version: ipv6
  # Allow heartbeat
  - chain: INPUT
    in_interface: "{{ octavia_container_interface }}"
    protocol: udp
    destination_port: "{{ octavia_health_manager_port }}"
    jump: ACCEPT
    ip_version: ipv6
  # Reject INPUT
  - chain: INPUT
    in_interface: "{{ octavia_container_interface }}"
    reject_with: icmp6-port-unreachable
    ip_version: ipv6
  # Reject FORWARD
  - chain: FORWARD
    in_interface: "{{ octavia_container_interface }}"
    reject_with: icmp6-port-unreachable
    ip_version: ipv6

# uWSGI Settings
octavia_wsgi_processes_max: 16
octavia_wsgi_processes: >-
  {{ [[(ansible_facts['processor_vcpus'] // ansible_facts['processor_threads_
↪per_core']) | default(1), 1] | max * 2, octavia_wsgi_processes_max] | min }}
octavia_wsgi_threads: 1
```

(continues on next page)

(continued from previous page)

```
octavia_uwsgi_bind_address: "{{ openstack_service_bind_address | default('0.0.
↪0.0') }}"
octavia_uwsgi_tls:
  crt: "{{ octavia_api_ssl_cert }}"
  key: "{{ octavia_api_ssl_key }}"

# Set up the drivers
# Provider agents are optional and not required for a successful Octavia␣
↪provider driver
# Possible options: amphora_agent, noop_agent, ovn
octavia_enabled_provider_agents:
  - "{{ (octavia_ovn_enabled | bool) | ternary('ovn', None) }}"

octavia_enabled_provider_drivers:
  - "amphora:'The Octavia Amphora driver.'"
  - "amphorav2:'The Octavia Amphora v2 driver.'"
  - "{{ (octavia_ovn_enabled | bool) | ternary(\"ovn:'The Octavia OVN␣
↪provider driver.'\", False) }}"
octavia_default_provider_driver: "amphorav2"
octavia_amphora_driver: amphora_haproxy_rest_driver
octavia_compute_driver: compute_nova_driver
octavia_network_driver: allowed_address_pairs_driver

# OVN Defaults
octavia_ovn_enabled: "{{ neutron_plugin_type | default('ml2.ovn') == 'ml2.ovn
↪' }}"
octavia_ovn_ssl: "{{ neutron_ovn_ssl | default(True) }}"
octavia_ovn_proto: "{{ (octavia_ovn_ssl) | ternary('ssl', 'tcp') }}"

octavia_ovn_nb_connection: >-
  {{ octavia_ovn_proto }}:{{ groups['neutron_ovn_northd'] | map('extract',␣
↪hostvars, ['ansible_host']) | join(':6641,' + octavia_ovn_proto + ':') }}
↪:6641
octavia_ovn_sb_connection: >-
  {{ octavia_ovn_proto }}:{{ groups['neutron_ovn_northd'] | map('extract',␣
↪hostvars, ['ansible_host']) | join(':6642,' + octavia_ovn_proto + ':') }}
↪:6642


#
# Certificate generation
#

# Set the host which will execute the openssl_* modules
# for the certificate generation. The host must already
# have access to pyOpenSSL.
octavia_cert_setup_host: "{{ openstack_pki_setup_host | default('localhost') }
↪}"
```

(continues on next page)

(continued from previous page)

```
# Set the directory where the certificates will be stored
# on the above host. If the host is localhost, then the user
# running the playbook must have access to it.
octavia_cert_dir: "{{ openstack_pki_dir | default(lookup('env', 'HOME',␣
↪default='/root') ~ '/openstack-ansible') }}"
octavia_cert_keys_dir: "{{ octavia_cert_dir }}/certs/private/"
octavia_cert_certs_dir: "{{ octavia_cert_dir }}/certs/certs/"
octavia_cert_dest_dir: "/etc/octavia/certs"

octavia_cert_client_req_common_name: "www.example.com" # change this to␣
↪something more real
octavia_cert_client_req_country_name: "US"
octavia_cert_client_req_state_or_province_name: "Denial"
octavia_cert_client_req_locality_name: "Nowhere"
octavia_cert_client_req_organization_name: "Dis"
octavia_cert_validity_days: 1825 # 5 years
octavia_generate_certs: true # generate self signed client certs
octavia_generate_client_cert: true
octavia_generate_ca: true
octavia_regenerate_client_cert: ""
octavia_regenerate_ca: ""

# OVN server certificate
# The local address used for the ovn certificate
octavia_ovn_node_address: "{{ management_address | default('127.0.0.1') }}"
# OVN destination files for SSL certificates
octavia_ovn_pki_intermediate_cert_name: "{{ octavia_api_intermediate_cert_
↪name }}"
octavia_ovn_pki_intermediate_chain_path: >-
  {{ octavia_cert_dir ~ '/roots/' ~ octavia_ovn_pki_intermediate_cert_name ~
↪'/certs/' ~ octavia_ovn_pki_intermediate_cert_name ~ '-chain.crt' }}
octavia_ovn_ssl_cert: "octavia_ovn.pem"
octavia_ovn_ssl_key: "octavia_ovn.key"
octavia_ovn_ssl_ca_cert: "octavia_ovn-ca.pem"

octavia_cert_authorities:
  - name: "OctaviaServerRoot"
    country: "{{ octavia_cert_client_req_country_name }}"
    state_or_province_name: "{{ octavia_cert_client_req_state_or_province_
↪name }}"
    organization_name: "{{ octavia_cert_client_req_organization_name }}"
    locality_name: "{{ octavia_cert_client_req_locality_name }}"
    cn: "Octavia Server CA"
    provider: selfsigned
    basic_constraints: "CA:TRUE"
    key_passphrase: "{{ octavia_ca_private_key_passphrase }}"
    key_usage:
      - digitalSignature
      - cRLSign
```

(continues on next page)

(continued from previous page)

```
      - keyCertSign
    not_after: "+{{ octavia_cert_validity_days }}d"
  - name: "OctaviaClientRoot"
    country: "{{ octavia_cert_client_req_country_name }}"
    state_or_province_name: "{{ octavia_cert_client_req_state_or_province_
↪name }}"
    organization_name: "{{ octavia_cert_client_req_organization_name }}"
    locality_name: "{{ octavia_cert_client_req_locality_name }}"
    cn: "Octavia Client CA"
    provider: selfsigned
    basic_constraints: "CA:TRUE"
    key_passphrase: "{{ octavia_cert_client_password }}"
    key_usage:
      - digitalSignature
      - cRLSign
      - keyCertSign
    not_after: "+{{ octavia_cert_validity_days }}d"

octavia_cert_certificates:
  # Communication between haproxy and octavia API
  - name: "octavia-api_{{ ansible_facts['hostname'] }}"
    provider: ownca
    cn: "{{ ansible_facts['hostname'] }}"
    san: "{{ octavia_api_cert_san }}"
    signed_by: "{{ octavia_api_intermediate_cert_name }}"
    condition: "{{ octavia_backend_ssl | bool }}"
  # Communication between octavia control plane and amphoras
  - name: "octavia_client"
    provider: ownca
    cn: "{{ octavia_cert_client_req_common_name }}"
    signed_by: "OctaviaClientRoot"
    ownca_key_passphrase: "{{ octavia_cert_client_password }}"
    key_usage:
      - nonRepudiation
      - digitalSignature
      - keyEncipherment
    extended_key_usage:
      - clientAuth
      - emailProtection
    condition: "{{ octavia_generate_certs | bool }}"
  # OVN NB/SB communication
  - name: "octavia_ovn_{{ ansible_facts['hostname'] }}"
    provider: ownca
    cn: "{{ ansible_facts['hostname'] }}"
    san: "{{ 'DNS:' ~ ansible_facts['hostname'] ~ ',IP:' ~ octavia_ovn_node_
↪address }}"
    signed_by: "{{ octavia_ovn_pki_intermediate_cert_name }}"
    condition: "{{ (octavia_ovn_ssl and octavia_ovn_enabled) }}"
```

(continues on next page)

(continued from previous page)

```
# Installation details for SSL certificates
octavia_cert_install_certificates:
  # Communication between haproxy and octavia API
  - src: "{{ octavia_api_user_ssl_cert | default(octavia_cert_certs_dir ~
↪'octavia-api_' ~ ansible_facts['hostname'] ~ '-chain.crt') }}"
    dest: "{{ octavia_api_ssl_cert }}"
    owner: "{{ octavia_system_user_name }}"
    group: "{{ octavia_system_user_name }}"
    mode: "0644"
    condition: "{{ octavia_backend_ssl | bool }}"
  - src: "{{ octavia_api_user_ssl_key | default(octavia_cert_keys_dir ~
↪'octavia-api_' ~ ansible_facts['hostname'] ~ '.key.pem') }}"
    dest: "{{ octavia_api_ssl_key }}"
    owner: "{{ octavia_system_user_name }}"
    group: "{{ octavia_system_user_name }}"
    mode: "0600"
    condition: "{{ octavia_backend_ssl | bool }}"
  # Server CA
  - src: "{{ octavia_ca_certificate | default(octavia_cert_dir ~ '/roots/
↪OctaviaServerRoot/certs/OctaviaServerRoot.crt') }}"
    dest: "{{ octavia_cert_dest_dir }}/server_ca.pem"
    owner: "{{ octavia_system_user_name }}"
    group: "{{ octavia_system_group_name }}"
    mode: "0640"
    condition: "{{ octavia_generate_certs | bool }}"
  - src: "{{ octavia_ca_private_key | default(octavia_cert_dir ~ '/roots/
↪OctaviaServerRoot/private/OctaviaServerRoot.key.pem') }}"
    dest: "{{ octavia_cert_dest_dir }}/ca_key.pem"
    owner: "{{ octavia_system_user_name }}"
    group: "{{ octavia_system_group_name }}"
    mode: "0640"
    condition: "{{ octavia_generate_certs | bool }}"
  # Client CA
  - src: "{{ octavia_client_ca | default(octavia_cert_dir ~ '/roots/
↪OctaviaClientRoot/certs/OctaviaClientRoot.crt') }}"
    dest: "{{ octavia_cert_dest_dir }}/client_ca.pem"
    owner: "{{ octavia_system_user_name }}"
    group: "{{ octavia_system_group_name }}"
    mode: "0640"
    condition: "{{ octavia_generate_certs | bool }}"
  # Client certificate
  - src: "{{ octavia_client_cert | default(octavia_cert_certs_dir ~ '/octavia_
↪client.crt') }}"
    dest: "{{ octavia_cert_dest_dir }}/client.pem.crt"
    owner: "{{ octavia_system_user_name }}"
    group: "{{ octavia_system_group_name }}"
    mode: "0640"
    condition: "{{ octavia_generate_certs | bool }}"
  - src: "{{ octavia_client_key | default(octavia_cert_keys_dir ~ '/octavia_
```

(continues on next page)

(continued from previous page)

```
↪client.key.pem') }}"
    dest: "{{ octavia_cert_dest_dir }}/client.pem.key"
    owner: "{{ octavia_system_user_name }}"
    group: "{{ octavia_system_group_name }}"
    mode: "0640"
    condition: "{{ octavia_generate_certs | bool }}"
  # OVN certificates
  - src: "{{ octavia_ovn_user_ssl_cert | default(octavia_cert_certs_dir ~
↪'octavia_ovn_' ~ ansible_facts['hostname'] ~ '-chain.crt') }}"
    dest: "{{ [octavia_cert_dest_dir, octavia_ovn_ssl_cert] | join('/') }}"
    owner: "{{ octavia_system_user_name }}"
    group: "{{ octavia_system_group_name }}"
    mode: "0644"
    condition: "{{ (octavia_ovn_ssl and octavia_ovn_enabled) }}"
  - src: "{{ octavia_ovn_user_ssl_key | default(octavia_cert_keys_dir ~
↪'octavia_ovn_' ~ ansible_facts['hostname'] ~ '.key.pem') }}"
    dest: "{{ [octavia_cert_dest_dir, octavia_ovn_ssl_key] | join('/') }}"
    owner: "{{ octavia_system_user_name }}"
    group: "{{ octavia_system_group_name }}"
    mode: "0600"
    condition: "{{ (octavia_ovn_ssl and octavia_ovn_enabled) }}"
  - src: "{{ octavia_ovn_user_ssl_ca_cert | default(octavia_ovn_pki_
↪intermediate_chain_path) }}"
    dest: "{{ [octavia_cert_dest_dir, octavia_ovn_ssl_ca_cert] | join('/') }}"
    owner: "{{ octavia_system_user_name }}"
    group: "{{ octavia_system_group_name }}"
    mode: "0644"
    condition: "{{ (octavia_ovn_ssl and octavia_ovn_enabled) }}"

# Custom client CA
# octavia_client_ca: "{{ octavia_cert_dir }}/ca_01.pem"
## Custom client certs
# octavia_client_cert: "{{ octavia_cert_dir }}/client.pem"
# octavia_client_key: "{{ octavia_cert_dir }}/client.key.pem"
## server
# octavia_server_ca: "{{ octavia_ca_certificate }}"
## ca certs
# octavia_ca_private_key: "{{ octavia_cert_dir }}/private/cakey.pem"
octavia_ca_private_key_passphrase: "{{ octavia_cert_client_password }}"
# octavia_ca_certificate: "{{ octavia_cert_dir }}/ca_server_01.pem"
# Custom OVN certs
# octavia_ovnnb_user_ssl_cert: <path to cert on ansible deployment host>
# octavia_ovnnb_user_ssl_key: <path to cert on ansible deployment host>
# octavia_ovnsb_user_ssl_cert: <path to cert on ansible deployment host>
# octavia_ovnsb_user_ssl_key: <path to cert on ansible deployment host>

## Tunable overrides
octavia_octavia_conf_overrides: {}
octavia_api_paste_ini_overrides: {}
```

(continues on next page)

(continued from previous page)

```
octavia_policy_overrides: {}
octavia_api_uwsgi_ini_overrides: {}

###
### Backend TLS
###

# Define if communication between haproxy and service backends should be
# encrypted with TLS.
octavia_backend_ssl: "{{ openstack_service_backend_ssl | default(False) }}"

# octavia server certificate
octavia_api_intermediate_cert_name: "{{ openstack_pki_service_intermediate_
↪cert_name | default('ExampleCorpIntermediate') }}"
octavia_api_cert_san: "{{ openstack_pki_san | default('DNS:' ~ ansible_facts[
↪'hostname'] ~ ',IP:' ~ management_address) }}"

# octavia destination files for SSL certificates
octavia_api_ssl_cert: "{{ octavia_cert_dest_dir }}/octavia-api.pem"
octavia_api_ssl_key: "{{ octavia_cert_dest_dir }}/octavia-api.key"

# Define user-provided SSL certificates
# octavia_api_user_ssl_cert: <path to cert on ansible deployment host>
# octavia_api_user_ssl_key: <path to cert on ansible deployment host>
```

CHAPTER

FOUR

EXAMPLE PLAYBOOK

```
---
- name: Install octavia server
  hosts: octavia_all
  user: root
  roles:
    - role: os_octavia
      tags:
        - os-octavia
  vars:
    external_lb_vip_address: 172.16.24.1
    internal_lb_vip_address: 192.168.0.1
    octavia_galera_address: "{{ internal_lb_vip_address }}"
    keystone_admin_user_name: admin
    keystone_admin_tenant_name: admin
```

4.1 Tags

This role supports the `octavia-install` and `octavia-config` tags. Use the `octavia-install` tag to install and upgrade. Use the `octavia-config` tag to maintain configuration of the service.