OpenStack-Ansible Documentation: os_octavia role

Release 18.1.0.dev244

OpenStack-Ansible Contributors

Apr 09, 2021

CONTENTS

CHAPTER

ONE

CONFIGURING THE OCTAVIA LOAD BALANCING SERVICE (OPTIONAL)

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 virtual machine 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 playbooks/containers-lxc-create.yml --limit lxc_
↪hosts,octavia_all
```

3. Define required overrides of the variables in defaults/main.yml of the openstack-ansible octavia role.
4. Run the os-octavia playbook

```
openstack-ansible playbooks/os-octavia-install.yml
```

5. Run the haproxy-install.yml playbook to add the new octavia API endpoints to the load balancer.

1.2 Setup a neutron network for use by octavia

Octavia needs connectivity between the control plane and the load balancing VMs. 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 VMs. 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 VM
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.

1.2.1 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:
     - neutron_linuxbridge_agent
     - octavia-worker
     - octavia-housekeeping
     - octavia-health-manager
```

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.

1.2.2 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"
    group_binds:
      - neutron_linuxbridge_agent
      - octavia-worker
      - octavia-housekeeping
      - octavia-health-manager
```

Add extend `user_variables.yml` with following overrides:

```
octavia_provider_network_name: vlan
octavia_provider_network_type: vlan
octavia_provider_segmentation_id: 400
octavia_container_network_name: lbaas_address
```

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.

1.3 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 uuid-runtime curl kpartx git jq python-pip
pip install virtualenv

virtualenv /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.git

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

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 abpout the diskimage script and the process at https://opendev.org/openstack/octavia/tree/master/diskimage-create

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

```
#/bin/sh

apt-get install qemu uuid-runtime curl kpartx git jq
pip -v >/dev/null || {apt-get install python-pip}
pip install virtualenv
virtualenv /opt/octavia-image-build || exit 1
source /opt/octavia-image-build/bin/activate

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

pushd octavia/diskimage-create
./diskimage-create.sh
mv amphora-x64-haproxy.qcow2 /tmp
deactivate

popd
popd

# upload image
openstack image delete amphora-x64-haproxy
```

(continues on next page)

(continued from previous page)

```
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
```

Note: If you have trouble installing dib-utils from pipy consider installing it directly from source *pip install git+https://opendev.org/openstack/dib-utils.git*

1.4 Creating the cryptographic certificates

Note: For production installation make sure that you review this very carefully with your own security requirements and potantially 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.

1.5 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 }}`

1.6 Optional: Tuning Octavia for production use

Please have a close look at the `main.yml` for tunable parameters. The most important change is to set Octavia into ACTIVE_STANDBY mode by adding `octavia_loadbalancer_topology: ACTIVE_STANDBY` and `octavia_enable_anti_affinity=True` to ensure that the active and passive amphora are (depending on the anti-affinity filter deployed in nova) on two different hosts to the user file in /etc/openstack-deploy

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

This is an OpenStack-Ansible role to deploy the Octavia Load Balancing service. See the role-octavia spec for more information.

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

CHAPTER

TWO

DEFAULT VARIABLES

```
## 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_python[
↪'executable'])) }}"

# Set installation method.
octavia_install_method: "source"
octavia_venv_python_executable: "{{ openstack_venv_python_executable |␣
↪default('python2') }}"

## Allow TLS listener
octavia_tls_listener_enabled: True

# Legacy policy disables the requirement for load-balancer service users to
# have one of the load-balancer:* roles.  It provides a similar policy to
# legacy OpenStack policies where any user or admin has access to load-
↪balancer
# resources that they own.  Users with the admin role has access to all
# load-balancer resources, whether they own them or not.
octavia_legacy_policy: False

# Set the package install state for distribution and pip packages
# Options are 'present' and 'latest'
octavia_package_state: "latest"
octavia_pip_package_state: "latest"

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_git_constraints:
  - "git+{{ octavia_git_repo }}@{{ octavia_git_install_branch }}
↪#egg=octavia"
  - "--constraint {{ octavia_upper_constraints_url }}"
```

(continues on next page)

(continued from previous page)

```
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: internalURL

octavia_auth_strategy: keystone

## Barbican certificates
octavia_barbican_enabled: false

## Cinder Volume
octavia_cinder_enabled: False

## 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_python['executable'])) }}"
octavia_galera_address: "{{ galera_address | default('127.0.0.1') }}"
octavia_galera_user: octavia
octavia_galera_database: octavia
octavia_galera_use_ssl: "{{ galera_use_ssl | default(False) }}"
octavia_galera_ssl_ca_cert: "{{ galera_ssl_ca_cert | default('/etc/ssl/
↪certs/galera-ca.pem') }}"
octavia_db_max_overflow: 20
octavia_db_pool_size: 120
octavia_db_pool_timeout: 30
octavia_galera_port: 3306

## 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_vhost: /octavia

# Notify
octavia_oslomsg_notify_host_group: "{{ oslomsg_notify_host_group | default(
↪'rabbitmq_all') }}"
octavia_oslomsg_notify_setup_host: "{{ (octavia_oslomsg_notify_host_group␣
↪in groups) | ternary(groups[octavia_oslomsg_notify_host_group][0],
↪'localhost') }}"
```

(continues on next page)

(continued from previous page)

```
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 }}"

## (Qdrouterd) integration
# TODO(ansmith): Change structure when more backends will be supported
octavia_oslomsg_amqp1_enabled: "{{ octavia_oslomsg_rpc_transport == 'amqp'␣
↪}}"

octavia_ceilometer_enabled: False

## 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 }}"

## Auth
octavia_service_region: RegionOne
octavia_service_project_name: "service"
octavia_service_user_name: "octavia"
octavia_service_role_name: admin
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: false
```

(continues on next page)

(continued from previous page)

```
## 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

# Common pip packages
octavia_pip_packages:
  - cryptography
  - keystonemiddleware
  - osprofiler
  - PyMySQL
  - pymemcache
  - python-glanceclient
  - python-keystoneclient
  - python-memcached
  - python-neutronclient
  - python-novaclient
  - python-openstackclient
  - python-octaviaclient
  - octavia
  - shade
  - systemd-python

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

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

octavia_optional_oslomsg_amqp1_pip_packages:
  - oslo.messaging[amqp1]

octavia_api_init_overrides: {}
octavia_worker_init_overrides: {}
octavia_housekeeping_init_overrides: {}
octavia_health_manager_init_overrides: {}

## 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_name: octavia-wsgi
    uwsgi_overrides: "{{ octavia_api_uwsgi_ini_overrides }}"
    uwsgi_port: "{{ octavia_service_port }}"
    uwsgi_bind_address: "{{ octavia_uwsgi_bind_address }}"
```

(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"

# 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
# ACTIVE_STANDBY is recommended for production settings
octavia_loadbalancer_topology: SINGLE

# 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
# 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-bionic.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') }}/openstack-ansible/
 ↪octavia"
octavia_amp_image_path_owner: "{{ lookup('env', 'USER') }}"
# enable uploading image to glance automatically
octavia_amp_image_upload_enabled: "{{ octavia_download_artefact }}"

# Name of the Octavia security group
```

(continues on next page)

(continued from previous page)

```
octavia_security_group_name: octavia_sec_grp
# Restrict access to only authorized hosts
octavia_security_group_rule_cidr:
# ssh enabled - switch to True if you need ssh access to the amphora
octavia_ssh_enabled: False
octavia_ssh_key_name: octavia_key
octavia_keypair_setup_host: "{{ openstack_service_setup_host | default(
↪'localhost') }}"
octavia_keypair_setup_host_python_interpreter: "{{ openstack_service_setup_
↪host_python_interpreter | default((octavia_keypair_setup_host ==
↪'localhost') | ternary(ansible_playbook_python, ansible_python[
↪'executable'])) }}"
# port the agent listens on
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: 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

# For convenience, we make use of the neutron message vhost
# for the LBAAS event stream. We could make a completely
# seperate vhost, but the old LBAAS functionality is
# deprecated so it's a bit pointless unless there is demand
# for doing it. As such, we provide these defaults here for
# the template to use.
neutron_oslomsg_rpc_userid: neutron
neutron_oslomsg_rpc_vhost: /neutron
neutron_oslomsg_rpc_transport: "{{ octavia_oslomsg_rpc_transport }}"
neutron_oslomsg_rpc_port: "{{ octavia_oslomsg_rpc_port }}"
neutron_oslomsg_rpc_servers: "{{ octavia_oslomsg_rpc_servers }}"
neutron_oslomsg_rpc_use_ssl: "{{ octavia_oslomsg_rpc_use_ssl }}"

# For additional security use a different user on the Neutron queue
# for Octavia with restricted access to only the event streamer
# queues
octavia_neutron_oslomsg_rpc_userid: "{{ neutron_oslomsg_rpc_userid }}"
octavia_neutron_oslomsg_rpc_password: "{{ neutron_oslomsg_rpc_password }}"

# 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).
# Since AIO only has one compute host this is set to false
octavia_enable_anti_affinity: False
```

(continues on next page)

(continued from previous page)

```
# 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:

# 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 provider net in the system
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
# 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: "True"
# Should Octavia set up the network and subnet?
octavia_service_net_setup: True
# This sets it to the container managment network based on how you setup
# the provider net
octavia_provider_network: "{{ provider_networks|map(attribute='network
↪')|selectattr('net_name','defined')|selectattr('net_name', 'equalto',␣
↪octavia_provider_network_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 heuritsics here but if you do anyhting special make␣
↪sure to use the
# ip addresses on the right network. This will use the container␣
↪newtorking 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 }}"
```

(continues on next page)

(continued from previous page)

```
# 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
  - # 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
```

(continues on next page)

(continued from previous page)

```
    ip_version: ipv6

# uWSGI Settings
octavia_wsgi_processes_max: 16
octavia_wsgi_processes: "{{ [[(ansible_processor_vcpus//ansible_processor_
↪threads_per_core)|default(1), 1] | max * 2, octavia_wsgi_processes_max]␣
↪| min }}"
octavia_wsgi_threads: 1
octavia_uwsgi_bind_address: "{{ openstack_service_bind_address | default(
↪'0.0.0.0') }}"
octavia_api_uwsgi_ini_overrides: {}

# Set up the drivers
octavia_amphora_driver: amphora_haproxy_rest_driver
octavia_compute_driver: compute_nova_driver
octavia_network_driver: allowed_address_pairs_driver

#
# 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_cert_setup_host | default('localhost
↪') }}"

# 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: "{{ lookup('env', 'HOME') }}/openstack-ansible/octavia"
octavia_cert_dir_owner: "{{ lookup('env', 'USER') }}"

octavia_cert_key_length_server: '4096' # key length
octavia_cert_cipher_server: 'aes256'
octavia_cert_cipher_client: 'aes256'
octavia_cert_key_length_client: '4096' # key length
octavia_cert_server_ca_subject: '/C=US/ST=Denial/L=Nowhere/O=Dis/CN=www.
↪example.com' # change this to something more real
octavia_cert_client_ca_subject: '/C=US/ST=Denial/L=Nowhere/O=Dis/CN=www.
↪example.com' # change this to something more real
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_client_cert: True # generate self signed client certs
octavia_generate_certs: True

# client certs
octavia_client_ca_key: "{{ octavia_cert_dir }}/ca_01.key"
octavia_client_ca: "{{ octavia_cert_dir }}/ca_01.pem"
octavia_client_cert: "{{ octavia_cert_dir }}/client.pem"
# server
```

(continues on next page)

(continued from previous page)

```
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"
octavia_signing_digest: sha256

# Quotas for the Octavia user - assuming active/passive topology
octavia_num_instances: 10000 # 5000 LB in active/passive
octavia_ram: "{{ (octavia_num_instances|int)*1024 }}"
octavia_num_server_groups: "{{ ((octavia_num_instances|int)*0.5)|int|abs }}
↪"
octavia_num_server_group_members: 50
octavia_num_cores: "{{ octavia_num_instances }}"
octavia_num_secgroups: "{{ (octavia_num_instances|int)*1.5|int|abs }}" #␣
↪average 3 listener per lb
octavia_num_ports: "{{ (octavia_num_instances|int)*10 }}" # at least␣
↪instances * 10
octavia_num_security_group_rules: "{{ (octavia_num_secgroups|int)*100 }}"

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

CHAPTER

THREE

DEPENDENCIES

This role needs pip >= 7.1 installed on the target 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.