OpenStack-Ansible Documentation: os_barbican role

Release 18.1.0.dev248

OpenStack-Ansible Contributors

Nov 28, 2024

CONTENTS

This Ansible role installs and configures OpenStack barbican.

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

CHAPTER

ONE

CONFIGURING THE KEY MANAGER (BARBICAN) SERVICE

First of all we need to set env.d parameters and define hosts where we want to run barbican-api. For that we can either extend *openstack_user_config.yml* or add file */etc/openstack_deploy/conf.d/barbican.yml* with the following content:

```
key-manager_hosts:
  infra1:
    ip: 172.29.236.11
  infra2:
    ip: 172.29.236.12
  infra3:
    ip: 172.29.236.13
```

Barbican can be configured to work in both single and multibackend modes. It depends on the number of keys in `barbican_backends_config` dictionary. Also you must explicitly define default backend in case variable contains configuration for more then one backend, for example:

```
barbican_backends_config:
  software:
    secret_store_plugin: store_crypto
    crypto_plugin: simple_crypto
  hsm:
    secret_store_plugin: store_crypto
    crypto_plugin: p11_crypto
    global_default: True
```

In addition to that you need to define plugin specific config, which can be done with definition of the variable `barbican_plugins_config`. Each key of this dictionary will be used to as config section name and values should be considered as key-value config options, like config_overrides options.

```
barbican_plugins_config:
  simple_crypto_plugin:
    kek: "{{ barbican_simple_crypto_key | b64encode }}"
```

Once all variables are set and configuration is done, you can deploy Barbican by running following playbooks:

```
# openstack-ansible playbooks/lxc-containers-create.yml --limit lxc_
↪hosts,barbican_all
# openstack-ansible playbooks/os-barbican-install.yml
# openstack-ansible playbooks/haproxy-install.yml
```

1.1 Configuring Barbican with Thales Luna HSM backend

As example we will show configuration for Thales Luna Network HSM (Safenet). Barbican stores in HSM only HMAC and MKEK keys, that are used to encrypt and decrypt keys that are stored in Barbican MySQL database.

MKEK is Master Key Encryption Key, which is used to encrypt KEKs that are unique and created per project. All keys within a project are encrypted with KEK.

You need to create a Data Protection On Demand sevice and complete an initialization of the Luna slot. You can follow lunacm documentation for more details. As a result of setup you should have:

1. Crypto Officer role password
2. Generated Chrystoki.conf
3. Binaries of libdpod.plugin and libCryptoki2.so

Note

At the moment barbican does not support Thales FIPS mode. Also mention, that Crypto Officer role password has to be reset on the first login, which should be done right after initialization of the CO role.

Once setup of the Thales is done, we can define all required variables that are needed for the Barbican deployment. Define in *user_variables.yml* following:

```
barbican_backends_config:
  hsm:
    secret_store_plugin: store_crypto
    crypto_plugin: p11_crypto

barbican_plugins_config:
  p11_crypto_plugin:
    library_path: /opt/barbican/libs/libCryptoki2.so
    login: "{{ barbican_dpod_co_password }}"
    slot_id: 3
    mkek_label: thales_mkek_3
    mkek_length: 32
    hmac_label: thales_hmac_3

barbican_user_libraries:
  - src: /etc/openstack_deploy/barbican/libCryptoki2.so
    dest: /opt/barbican/libs/libCryptoki2.so
  - src: /etc/openstack_deploy/barbican/libdpod.plugin
    dest: /opt/barbican/libs/plugins/libdpod.plugin
  - src: /etc/openstack_deploy/barbican/Chrystoki.conf
    dest: /opt/barbican/Chrystoki.conf
```

You should also add `barbican_dpod_co_password` to *user_secrets.yml* and set to the value of Crypto Officer role password.

We would need to symlink Chrystoki.conf so /etc. Additionally it is required to manually generate hmac and mkek keys, that would be stored on HSM.

```
# ansible -m file -a "src=/opt/barbican/Chrystoki.conf dest=/etc/
↪Chrystoki.conf state=link" barbican_all
# ansible -m command -a "/openstack/venvs/barbican-{{ venv_tag }}/
↪bin/barbican-manage hsm gen_hmac --library-path /opt/libs/64/
↪libCryptoki2.so --passphrase {{ barbican_dpod_co_password }} --
↪slot-id 3 --label thales_hmac_3" barbican_all[0]
# ansible -m command -a "/openstack/venvs/barbican-{{ venv_tag }}/
↪bin/barbican-manage hsm gen_mkek --library-path /opt/libs/64/
↪libCryptoki2.so --passphrase {{ barbican_dpod_co_password }} --
↪slot-id 3 --label thales_mkek_3" barbican_all[0]
```

1.2 Configuring Barbican with Entrust nShield Connect HSM backend

The following example demonstrates a configuration supporting the Entrust nShield Connect HSM. Barbican stores HMAC and MKEK keys in the HSM, which are used to encrypt and decrypt keys that are stored in Barbican MySQL database.

MKEK stands for **Master Key Encryption Key**, which is used to encrypt KEKs that are unique and created per project. All keys within a project are encrypted with KEK.

Before proceeding, you must install the Security World software provided by Entrust. The software will install libraries that will be referenced as part of the configuration. In addition, the HSM may utilize one or more slots that will also be required to complete the configuration. Please consult the nShield Connect User Guide for Linux and/or Entrust support for assistance.

Once the installation is complete, you should know or have:

1. Desired Slot ID
2. The `libcknfast.so` library file

The Slot ID can be determined using the `pcks11-tool` as shown here:

```
# pkcs11-tool -L --module /opt/nfast/toolkits/pkcs11/libcknfast.so
Available slots:
Slot 0 (0x1d622495): 6606-XXXX-XXXX Rt2
  token label        : accelerator
  token manufacturer : nCipher Corp. Ltd
  token model        :
  token flags        : rng, token initialized, other flags=0x200
  hardware version   : 0.12
  firmware version   : 12.50
  serial num         : 6606-XXXX-XXXX
  pin min/max        : 0/256
Slot 1 (0x1d622496): 6606-XXXX-XXXX Rt2 slot 0
  (token not recognized)
Slot 2 (0x1d622497): 6606-XXXX-XXXX Rt2 slot 2
  (empty)
Slot 3 (0x1d622498): 6606-XXXX-XXXX Rt2 slot 3
  (empty)
```

The usable slot value is in HEX and must be converted to decimal:

```
# echo $((0x1d622495))
492971157
```

Once the nShield-related setup is complete, we can define all required variables that are needed for the Barbican deployment. For convenience, copy the `libcknfast.so` library to `/etc/openstack_deploy/barbican/` on the deploy node. It will be distributed amongst the Barbican service nodes accordingly.

Define the following in *user_variables.yml*:

```
barbican_backends_config:
  hsm:
    secret_store_plugin: store_crypto
    crypto_plugin: p11_crypto

barbican_plugins_config:
  p11_crypto_plugin:
    library_path: /opt/barbican/libs/libcknfast.so
    token_serial_number: 12345678
    login: mypassword123
    slot_id: 492971157
    mkek_label: thales_mkek_0
    mkek_length: 32
    hmac_label: thales_hmac_0
    encryption_mechanism: CKM_AES_CBC
    hmac_key_type: CKK_SHA256_HMAC
    hmac_keygen_mechanism: CKK_SHA256_HMAC

barbican_user_libraries:
  - src: /etc/openstack_deploy/barbican/libcknfast.so
    dest: /opt/barbican/libs/libcknfast.so
```

Override variables can be added or modified as needed.

To generate the HMAC key, perform the following command using the approrpiate values:

```
barbican-manage hsm gen_hmac \
--library-path /opt/nfast/toolkits/pkcs11/libcknfast.so \
--passphrase mypassword123 --slot-id 492971157 --label thales_hmac_0␣
↪\
--key-type CKK_SHA256_HMAC \
--mechanism CKM_NC_SHA256_HMAC_KEY_GEN
```

To generate the MKEK key, perform the following command using the approrpiate values:

```
barbican-manage hsm gen_mkek \
--library-path /opt/nfast/toolkits/pkcs11/libcknfast.so \
--passphrase mypassword123 --slot-id 492971157 --label thales_mkek_0
```

Lastly, restart the nCipher service(s) and Barbican API service:

```
# /opt/nfast/sbin/init.d-ncipher restart
# systemctl restart barbican-api
```

1.3 Configuring Barbican with Vault backend

HashiCorp Vault is pretty popular key storage engine that is used in production by a lot of companies. You have 2 ways to use Vault with OpenStack:

1. Connect services directly to Vault with Castellan. In this case all keys would be stored inside Vault under the same user and no tenant isolation will be present. This option does not require Barbican deployment at all.
2. Connect services to Barbican, Barbican is connected to Vault. In this scenario we configure Castellan to use Barbican driver and services will reach it for the secrets. In this case Barbican will generate KEK which will be unique per project and store secrets inside its MySQL database. Master KEKs in their turn will be stored inside Vault.

In both options you would need to create a KV2 Secret Storage in Vault.

1.3.1 Connect services directly to Vault

Eventually this section is not related to Barbican at all, since it does not require Barbican endpoints to be present at all and it needs only services configuration (like Nova or Cinder). To use it you would need to define following overrides is *user_variables.yml*:

```
nova_nova_conf_overrides:
  key_manager:
    backend: vault
  vault:
    kv_mountpoint: secret
    root_token_id: "{{ vault_root_token }}"
    vault_url: https://vault.example.com
    use_ssl: True

cinder_cinder_conf_overrides:
  key_manager:
    backend: vault
  vault:
    kv_mountpoint: secret
    root_token_id: "{{ vault_root_token }}"
    vault_url: https://vault.example.com
    use_ssl: True
```

After variables are set we need to run roles to re-configure services:

```
# openstack-ansible playbooks/os-cinder-install.yml --tags cinder-
↪config
# openstack-ansible playbooks/os-nova-install.yml -- tags nova-config
```

1.3.2 Connect Barbican to Vault

You need to define variables like shown in the sample below to configure Babrican to use Vault store driver:

```
barbican_backends_config:
  vault:
    secret_store_plugin: vault_plugin
    crypto_plugin: simple_crypto

barbican_plugins_config:
  vault_plugin:
    kv_mountpoint: secret
    root_token_id: "{{ vault_root_token }}"
    vault_url: https://vault.example.com
    use_ssl: True
```

1.4 Configuring services to work with Barbican

We need to let know Cinder, Nova and other services that key storage (Barbican) is available now for interaction. There are special variables like `<service>_barbican_enabled` that should be set to True once there are at least one host in `barbican_all` group. So generally it should be enough just to re-run service-related roles to get services config adjusted to interact with barbican:

```
# openstack-ansible playbooks/os-cinder-install.yml --tags cinder-config
# openstack-ansible playbooks/os-nova-install.yml --tags nova-config
```

Then we can make use of barbican, for example, to make LUKS-encrypted volumes. You may reference to Cinder docs for sample usage.

You should also make sure, that tenants do have *creator* role assigned as it is required to be able to create secrets in Barbican.

CHAPTER

TWO

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.
barbican_service_setup_host: "{{ openstack_service_setup_host | default(
↪'localhost') }}"
barbican_service_setup_host_python_interpreter: >-
  {{
    openstack_service_setup_host_python_interpreter | default(
      (barbican_service_setup_host == 'localhost') | ternary(ansible_playbook_
↪python, ansible_facts['python']['executable']))
  }}

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

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

# Toggle keystone authentication for barbican
barbican_keystone_auth: "{{ (groups['keystone_all'] is defined) and (groups[
↪'keystone_all'] | length > 0) }}"

## System info
barbican_system_group_name: barbican
barbican_system_user_name: barbican
barbican_system_user_comment: Barbican System User
barbican_system_user_shell: /bin/false
barbican_system_user_home: "/var/lib/{{ barbican_system_user_name }}"
barbican_etc_directory: /etc/barbican

# Barbican services info
barbican_keystone_listener_enable: false
```

(continues on next page)

(continued from previous page)

```
barbican_worker_enable: false
barbican_retry_enable: false

# Variable defines barbican store backends configuration. It supports␣
↪multibackend scenario
# in case list length > 1. Then additional key global_default should be␣
↪present, otherwise
# first element would be set as global default. For multibackend one backend␣
↪should be set
# as global_default: True
barbican_backends_config:
  software:
    secret_store_plugin: store_crypto
    crypto_plugin: simple_crypto

# Variable defines barbican crypto configuration.
barbican_plugins_config:
  simple_crypto_plugin:
    kek: "{{ barbican_simple_crypto_key | b64encode }}"

## Service Name-Group Mapping
barbican_services:
  barbican-api:
    group: barbican_all
    service_name: barbican-api
    init_config_overrides: "{{ barbican_init_config_overrides }}"
    uwsgi_bind_address: "{{ barbican_service_host }}"
    uwsgi_port: "{{ barbican_service_port }}"
    uwsgi_overrides: "{{ barbican_uwsgi_init_overrides }}"
    wsgi_app: True
    wsgi_name: barbican-wsgi-api
    start_order: 1
    uwsgi_tls: "{{ barbican_backend_ssl | ternary(barbican_uwsgi_tls, {}) }}"
  barbican-worker:
    group: barbican_all
    service_name: barbican-worker
    init_config_overrides: "{{ barbican_init_config_overrides }}"
    execstarts: "{{ barbican_bin }}/barbican-worker"
    condition: "{{ barbican_worker_enable | bool }}"
    start_order: 2
  barbican-keystone-listener:
    group: barbican_all
    service_name: barbican-keystone-listener
    init_config_overrides: "{{ barbican_init_config_overrides }}"
    execstarts: "{{ barbican_bin }}/barbican-keystone-listener"
    condition: "{{ barbican_keystone_listener_enable | bool }}"
    start_order: 3
  barbican-retry:
    group: barbican_all
```

(continues on next page)

(continued from previous page)

```
    service_name: barbican-retry
    init_config_overrides: "{{ barbican_init_config_overrides }}"
    execstarts: "{{ barbican_bin }}/barbican-retry"
    condition: "{{ barbican_retry_enable | bool }}"
    start_order: 4


# With `barbican_user_libraries` you can deploy libraries, needed for barbican
# to interact with third party services like HSM
# barbican_user_libraries:
#   - src: /etc/openstack_deploy/barbican/libdpod.plugin
#     dest: /opt/barbican/libs/libCryptoki2.so
#     owner: root
#     group: "{{ barbican_system_group_name }}"
#   - src: /etc/openstack_deploy/barbican/Chrystoki.conf
#     dest: /opt/barbican/Chrystoki.conf
#     link: /etc/Chrystoki.conf

barbican_user_libraries: []

## Service Type and Data
barbican_service_name: barbican
barbican_service_user_name: barbican
barbican_service_type: key-manager
barbican_service_description: "OpenStack Key and Secrets Management (Barbican)
↪"
barbican_default_role_names:
  - "key-manager:service-admin"
  - creator
  - observer
  - audit
barbican_service_role_names:
  - admin
  - creator
  - service
barbican_service_token_roles:
  - service
barbican_service_token_roles_required: "{{ openstack_service_token_roles_
↪required | default(True) }}"
barbican_service_region: "{{ service_region | default('RegionOne') }}"
barbican_service_host: "{{ openstack_service_bind_address | default('0.0.0.0
↪') }}"
barbican_service_port: 9311
barbican_service_proto: http
barbican_service_publicuri_proto: "{{ openstack_service_publicuri_proto |␣
↪default(barbican_service_proto) }}"
barbican_service_adminuri_proto: "{{ openstack_service_adminuri_proto |␣
↪default(barbican_service_proto) }}"
barbican_service_internaluri_proto: "{{ openstack_service_internaluri_proto |␣
```

(continues on next page)

(continued from previous page)

```
↪default(barbican_service_proto) }}"
barbican_service_publicurl: "{{ barbican_service_publicuri_proto }}://{{␣
↪external_lb_vip_address }}:{{ barbican_service_port }}"
barbican_service_internalurl: "{{ barbican_service_internaluri_proto }}://{{␣
↪internal_lb_vip_address }}:{{ barbican_service_port }}"
barbican_service_adminurl: "{{ barbican_service_adminuri_proto }}://{{␣
↪internal_lb_vip_address }}:{{ barbican_service_port }}"

barbican_service_in_ldap: "{{ service_ldap_backend_enabled | default(False) }}
↪"

barbican_init_config_overrides: {}
barbican_config_overrides: {}
barbican_policy_overrides: {}
barbican_paste_overrides: {}
barbican_api_audit_map_overrides: {}
barbican_vassals_api_overrides: {}

## The git source/branch
barbican_git_repo: "https://opendev.org/openstack/barbican"
barbican_git_install_branch: master
barbican_upper_constraints_url: >-
  {{ requirements_git_url | default('https://releases.openstack.org/
↪constraints/upper/' ~ requirements_git_install_branch | default('master')) }
↪}
barbican_git_constraints:
  - "--constraint {{ barbican_upper_constraints_url }}"

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

# Name of the virtual env to deploy into
barbican_venv_tag: "{{ venv_tag | default('untagged') }}"
barbican_bin: "{{ _barbican_bin }}"

# Database vars
barbican_db_setup_host: "{{ openstack_db_setup_host | default('localhost') }}"
barbican_db_setup_python_interpreter: >-
  {{
    openstack_db_setup_python_interpreter | default(
      (barbican_db_setup_host == 'localhost') | ternary(ansible_playbook_
↪python, ansible_facts['python']['executable']))
  }}
barbican_galera_address: "{{ galera_address | default('127.0.0.1') }}"
barbican_galera_database: barbican
barbican_galera_user: barbican
barbican_galera_use_ssl: "{{ galera_use_ssl | default(False) }}"
barbican_galera_ssl_ca_cert: "{{ galera_ssl_ca_cert | default('') }}"
barbican_galera_port: "{{ galera_port | default('3306') }}"
# NOTE: barbican does not support pool_timeout so it is not set for this role
```

(continues on next page)

(continued from previous page)

```
barbican_db_max_overflow: "{{ openstack_db_max_overflow | default('50') }}"
barbican_db_max_pool_size: "{{ openstack_db_max_pool_size | default('5') }}"
barbican_db_pool_timeout: "{{ openstack_db_pool_timeout | default('30') }}"
barbican_db_connection_recycle_time: "{{ openstack_db_connection_recycle_time␣
↪| default('600') }}"

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

# RPC
barbican_oslomsg_rpc_host_group: "{{ oslomsg_rpc_host_group | default(
↪'rabbitmq_all') }}"
barbican_oslomsg_rpc_setup_host: "{{ (barbican_oslomsg_rpc_host_group in␣
↪groups) | ternary(groups[barbican_oslomsg_rpc_host_group][0], 'localhost') }
↪}"
barbican_oslomsg_rpc_transport: "{{ oslomsg_rpc_transport | default('rabbit')␣
↪}}"
barbican_oslomsg_rpc_servers: "{{ oslomsg_rpc_servers | default('127.0.0.1') }
↪}"
barbican_oslomsg_rpc_port: "{{ oslomsg_rpc_port | default('5672') }}"
barbican_oslomsg_rpc_use_ssl: "{{ oslomsg_rpc_use_ssl | default(False) }}"
barbican_oslomsg_rpc_userid: barbican
barbican_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 `/`.
barbican_oslomsg_rpc_vhost:
  - name: /barbican
    state: "{{ barbican_oslomsg_rabbit_quorum_queues | ternary('absent',
↪'present') }}"
  - name: barbican
    state: "{{ barbican_oslomsg_rabbit_quorum_queues | ternary('present',
↪'absent') }}"
barbican_oslomsg_rpc_ssl_version: "{{ oslomsg_rpc_ssl_version | default(
↪'TLSv1_2') }}"
barbican_oslomsg_rpc_ssl_ca_file: "{{ oslomsg_rpc_ssl_ca_file | default('') }}
↪"

# Notify
barbican_oslomsg_notify_configure: "{{ oslomsg_notify_configure |␣
↪default(barbican_ceilometer_enabled) }}"
barbican_oslomsg_notify_host_group: "{{ oslomsg_notify_host_group | default(
↪'rabbitmq_all') }}"
barbican_oslomsg_notify_setup_host: "{{ (barbican_oslomsg_notify_host_group␣
↪in groups) | ternary(groups[barbican_oslomsg_notify_host_group][0],
↪'localhost') }}"
barbican_oslomsg_notify_transport: "{{ oslomsg_notify_transport | default(
↪'rabbit') }}"
```

(continues on next page)

(continued from previous page)

```
barbican_oslomsg_notify_servers: "{{ oslomsg_notify_servers | default('127.0.
↪0.1') }}"
barbican_oslomsg_notify_port: "{{ oslomsg_notify_port | default('5672') }}"
barbican_oslomsg_notify_use_ssl: "{{ oslomsg_notify_use_ssl | default(False) }
↪}"
barbican_oslomsg_notify_userid: "{{ barbican_oslomsg_rpc_userid }}"
barbican_oslomsg_notify_password: "{{ barbican_oslomsg_rpc_password }}"
barbican_oslomsg_notify_vhost: "{{ barbican_oslomsg_rpc_vhost }}"
barbican_oslomsg_notify_ssl_version: "{{ oslomsg_notify_ssl_version | default(
↪'TLSv1_2') }}"
barbican_oslomsg_notify_ssl_ca_file: "{{ oslomsg_notify_ssl_ca_file | default(
↪'') }}"
barbican_oslomsg_notify_policies: []

## RabbitMQ integration
barbican_oslomsg_rabbit_quorum_queues: "{{ oslomsg_rabbit_quorum_queues |␣
↪default(True) }}"
barbican_oslomsg_rabbit_stream_fanout: "{{ oslomsg_rabbit_stream_fanout |␣
↪default(barbican_oslomsg_rabbit_quorum_queues) }}"
barbican_oslomsg_rabbit_transient_quorum_queues: "{{ oslomsg_rabbit_transient_
↪quorum_queues | default(barbican_oslomsg_rabbit_stream_fanout) }}"
barbican_oslomsg_rabbit_qos_prefetch_count: "{{ oslomsg_rabbit_qos_prefetch_
↪count | default(barbican_oslomsg_rabbit_stream_fanout | ternary(10, 0)) }}"
barbican_oslomsg_rabbit_queue_manager: "{{ oslomsg_rabbit_queue_manager |␣
↪default(barbican_oslomsg_rabbit_quorum_queues) }}"
barbican_oslomsg_rabbit_quorum_delivery_limit: "{{ oslomsg_rabbit_quorum_
↪delivery_limit | default(0) }}"
barbican_oslomsg_rabbit_quorum_max_memory_bytes: "{{ oslomsg_rabbit_quorum_
↪max_memory_bytes | default(0) }}"

## (Qdrouterd) integration
# TODO(ansmith): Change structure when more backends will be supported
barbican_oslomsg_amqp1_enabled: "{{ barbican_oslomsg_rpc_transport == 'amqp' }
↪}"

# Keystone AuthToken/Middleware
barbican_keystone_auth_plugin: password
barbican_service_project_domain_id: default
barbican_service_user_domain_id: default
barbican_service_project_name: service

# uwsgi configuration vars
barbican_wsgi_processes_max: 16
barbican_wsgi_processes: >-
  {{ [[(ansible_facts['processor_vcpus'] // ansible_facts['processor_threads_
↪per_core']) | default(1), 1] | max * 2, barbican_wsgi_processes_max] | min }
↪}
barbican_wsgi_threads: 1
barbican_uwsgi_tls:
```

(continues on next page)

(continued from previous page)

```
  crt: "{{ barbican_ssl_cert }}"
  key: "{{ barbican_ssl_key }}"

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

# packages required to run the barbican service
barbican_pip_packages:
  - "git+{{ barbican_git_repo }}@{{ barbican_git_install_branch }}
↪#egg=barbican"
  - osprofiler
  - PyMySQL
  - pymemcache
  - python-memcached
  - systemd-python
barbican_user_pip_packages: []

barbican_optional_oslomsg_amqp1_pip_packages:
  - oslo.messaging[amqp1]

barbican_uwsgi_init_overrides: {}

###
### Backend TLS
###

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

# Storage location for SSL certificate authority
barbican_pki_dir: "{{ openstack_pki_dir | default('/etc/openstack_deploy/pki
↪') }}"

# Delegated host for operating the certificate authority
barbican_pki_setup_host: "{{ openstack_pki_setup_host | default('localhost') }
↪}"

# barbican server certificate
barbican_pki_keys_path: "{{ barbican_pki_dir ~ '/certs/private/' }}"
barbican_pki_certs_path: "{{ barbican_pki_dir ~ '/certs/certs/' }}"
barbican_pki_intermediate_cert_name: "{{ openstack_pki_service_intermediate_
↪cert_name | default('ExampleCorpIntermediate') }}"
barbican_pki_regen_cert: ''
barbican_pki_san: "{{ openstack_pki_san | default('DNS:' ~ ansible_facts[
↪'hostname'] ~ ',IP:' ~ management_address) }}"
barbican_pki_certificates:
  - name: "barbican_{{ ansible_facts['hostname'] }}"
    provider: ownca
```

(continues on next page)

(continued from previous page)

```
    cn: "{{ ansible_facts['hostname'] }}"
    san: "{{ barbican_pki_san }}"
    signed_by: "{{ barbican_pki_intermediate_cert_name }}"

# barbican destination files for SSL certificates
barbican_ssl_cert: /etc/barbican/barbican.pem
barbican_ssl_key: /etc/barbican/barbican.key

# Installation details for SSL certificates
barbican_pki_install_certificates:
  - src: "{{ barbican_user_ssl_cert | default(barbican_pki_certs_path ~
↪'barbican_' ~ ansible_facts['hostname'] ~ '-chain.crt') }}"
    dest: "{{ barbican_ssl_cert }}"
    owner: "{{ barbican_system_user_name }}"
    group: "{{ barbican_system_user_name }}"
    mode: "0644"
  - src: "{{ barbican_user_ssl_key | default(barbican_pki_keys_path ~
↪'barbican_' ~ ansible_facts['hostname'] ~ '.key.pem') }}"
    dest: "{{ barbican_ssl_key }}"
    owner: "{{ barbican_system_user_name }}"
    group: "{{ barbican_system_user_name }}"
    mode: "0600"

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

CHAPTER

THREE

DEPENDENCIES

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

This role requires the following variables to be defined:

```
barbican_galera_address
barbican_galera_password
barbican_oslomsg_rpc_password
barbican_service_password
keystone_admin_user_name
keystone_auth_admin_password
keystone_admin_tenant_name
```

CHAPTER

FOUR

EXAMPLE PLAYBOOK

```yaml
---
- name: Install barbican server
  hosts: barbican_all
  user: root
  roles:
    - role: "os_barbican"
  vars:
    external_lb_vip_address: 172.16.24.1
    internal_lb_vip_address: 192.168.0.1
    barbican_galera_address: "{{ internal_lb_vip_address }}"
    barbican_service_password: SuperSecretePassword1
    barbican_galera_password: SuperSecretePassword2
    barbican_oslomsg_rpc_password: SuperSecretePassword3
    barbican_oslomsg_notify_password: "{{ barbican_oslomsg_rpc_password }}" #␣
↪if using the same user, please use the same password
    keystone_admin_user_name: admin
    keystone_auth_admin_password: SuperSecretePassword5
    keystone_admin_tenant_name: admin
    galera_root_user: root
  vars_prompt:
    - name: "galera_root_password"
      prompt: "What is galera_root_password?"
```

CHAPTER

FIVE

TAGS

This role supports two tags: `barbican-install` and `barbican-config`. The `barbican-install` tag can be used to install and upgrade. The `barbican- config` tag can be used to maintain configuration of the service.