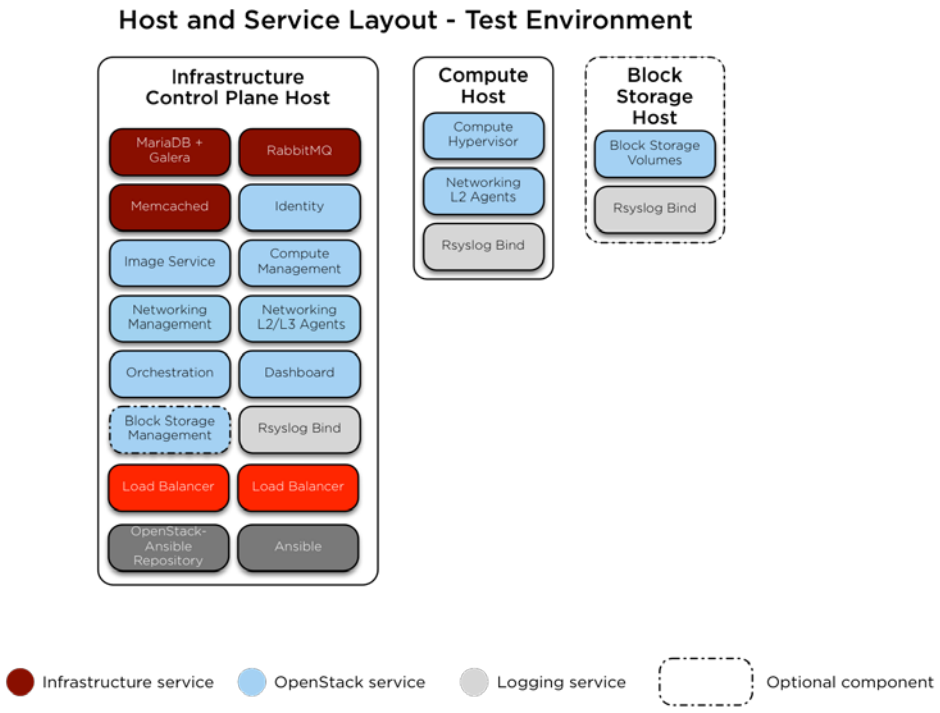
Appendix A: Example test environment configuration

A test environment contains the minimal set of components needed to deploy a working OpenStack-Ansible (OSA) environment for testing purposes.

A test environment has the following characteristics:

- One infrastructure (control plane) host (8 vCPU, 8 GB RAM, 60 GB HDD)
- One compute host (8 vCPU, 8 GB RAM, 60 GB HDD)
- One Network Interface Card (NIC) for each host
- A basic compute kit environment with the Image (glance) and Compute (nova) services set to use file-backed storage.



Deleted: ¶
Introduction

Deleted: The

Deleted: is

Deleted: a

Deleted: The

Deleted: 8GB

Deleted: 60GB

Deleted: 8GB

Deleted: 60GB

Deleted: Each host only has o

Deleted: Only a

Deleted: will be installed

Network configuration

[If you add a code example here, be sure to also provide a text introduction for it.]

[Code block that is TBD]

Environment configuration

The `/etc/openstack_deploy/openstack_user_config.yml` configuration file defines which hosts run the containers and services deployed by OSA. For example, hosts listed in the `shared-infra_hosts` section run containers for many of the shared services that your OpenStack environment requires. Following is an example of the `/etc/openstack_deploy/openstack_user_config.yml` configuration file for a test environment.

[Code block that I didn't copy]

Deleted: sets

Deleted: the hosts available in the groups. This designates the services that runs on them.

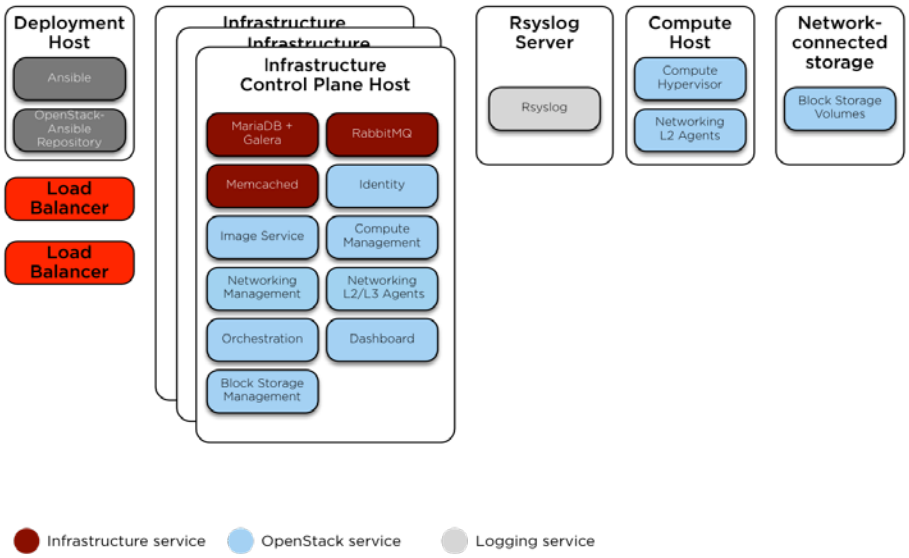
Appendix B: Example production environment configuration

A production environment contains the minimal set of components needed to deploy a working OpenStack-Ansible (OSA) environment for production purposes.

A production environment has the following characteristics:

- Three infrastructure (control plane) hosts
- Two compute hosts
- One storage host
- One log aggregation host
- Two network agent hosts
- Multiple Network Interface Cards (NIC) configured as bonded pairs for each host
- Full compute kit with the Telemetry service (ceilometer) included, with NFS configured as a storage back end for the Compute (nova), Image (glance), and Block Storage (cinder) services.

Host and Service Layout - Production Environment



Deleted: ¶
Introduction

Deleted: The

Deleted: is a

Deleted: The

Deleted: 3

Deleted: 2

Deleted: 1

Deleted: 1

Deleted: 2

Deleted: Each host m

Deleted: .

Deleted: The f

Deleted: will be installed

Deleted:

Deleted: -

Network configuration

[Insert an intro to the code block.]

[Code block that I didn't copy]

Environment configuration

The `/etc/openstack_deploy/openstack_user_config.yml` configuration file defines which hosts run the containers and services deployed by OSA. For example, hosts listed in the `shared-infra_hosts` section run containers for many of the shared services that your OpenStack environment requires. Following is an example of the `/etc/openstack_deploy/openstack_user_config.yml` configuration file for a production environment.

Deleted: sets the hosts available in the groups. This designates the services that runs on them

[Code block that I didn't copy]

Appendix C: Customizing host and service layouts

The default layout of containers and services in OpenStack-Ansible (OSA) is determined by the `/etc/openstack_deploy/openstack_user_config.yml` file and the contents of both the `/etc/openstack_deploy/conf.d/` and `/etc/openstack_deploy/env.d/` directories. You use these sources to define the *group* mappings that the playbooks use to target hosts and containers for roles used in the deployment.

- You define *host groups*, which gather the target hosts into *inventory groups*, through the `/etc/openstack_deploy/openstack_user_config.yml` file and the contents of the `/etc/openstack_deploy/conf.d/` directory.

- You define *container groups*, which can map from the service components to be deployed up to host groups, through files in the `/etc/openstack_deploy/env.d/` directory.

To customize the layout of the components for your deployment, modify the host groups and container groups appropriately before running the installation playbooks.

Understanding host groups

As part of the initial configuration, each target host appears either in the `/etc/openstack_deploy/openstack_user_config.yml` file or in files within the `/etc/openstack_deploy/conf.d/` directory. The format used for files in the `conf.d/` directory is identical to the syntax used in the `openstack_user_config.yml` file.

In these files, the target hosts are listed under one or more headings, such as `shared-infra_hosts` or `storage_hosts`, which serve as Ansible group mappings. These groups map to the physical hosts.

The `haproxy.yml.example` file in the `conf.d/` directory provides a simple example of defining a host group (`haproxy_hosts`) with two hosts (`infra1` and `infra2`).

The `swift.yml.example` file provides a more complex example. Here, host variables for a target host are specified by using the `container_vars` key. OSA applies all entries under this key as host-specific variables to any component containers on the specific host.

Deleted: ¶

¶

Understanding the default layout

Deleted: driven

Deleted: Use

Formatted: Font: Italic

Deleted: used by

Comment [KH1]: The end of this sentence doesn't quite make sense. I think there might be a verb missing after "to"?

Deleted: ¶

Conceptually, these can be thought of as mapping from two directions.

Formatted: List Paragraph, Bulleted + Level: 1 + Aligned at: 0.25" + Indent at: 0.5"

Formatted: Font: Italic

Formatted: Font: Italic

Formatted: Font: Italic

Deleted: to represent the layout you desire

Comment [KH2]: Bring this heading up a level.

Deleted: either

Deleted: We use a

Deleted: which

Comment [KH3]: Note that I suggest starting a new paragraph here.

Deleted: These

Deleted: We treat t

Deleted: ings as mappings

Deleted: example file

Deleted: A

Deleted: file is `swift.yml.example`

Deleted: we specify

Deleted: OpenStack-Ansible

Note: To manage file size, we recommend that you define new inventory groups, particularly for new services, by using a new file in the `conf.d/` directory.

Understanding container groups

Additional group mappings are located within files in the `/etc/openstack_deploy/env.d/` directory. These groups are treated as virtual mappings from the host groups to the container groups that define where each service deploys. By reviewing files within the `env.d/` directory, you can begin to see the nesting of groups represented in the default layout.

For example, the `shared-infra.yml` file defines a container group, `shared-infra_containers`, as a subset of the `all_containers` inventory group. The `shared-infra_containers` container group is mapped to the `shared-infra_hosts` host group. All of the service components in the `shared-infra_containers` container group are deployed to each target host in the `shared-infra_hosts` host group.

Within a `physical_skel` section, the OSA dynamic inventory expects to find a pair of keys. The first key maps to items in the `container_skel` section, and the second key maps to the target host groups that are responsible for hosting the service component.

To continue the example, the `memcache.yml` file defines the `memcache_container` container group. This group is a subset of the `shared-infra_containers` group, which is itself a subset of the `all_containers` inventory group.

Note: The `all_containers` group is automatically defined by OSA. Any service component managed by OSA maps to a subset of the `all_containers` inventory group, directly or indirectly through another intermediate container group.

The default layout does not rely exclusively on groups being subsets of other groups. The `memcache` component group is part of the `memcache_container` group, as well as the `memcache_all` group, and also contains a `memcached` component group. If you review the `playbooks/memcached-install.yml` playbook, you see that the playbook applies to hosts in the `memcached` group. Other services might have more complex deployment needs. They define and consume inventory container groups differently. Mapping components to several groups in this way allows flexible targeting of roles and tasks.

Deleted: W
Deleted: to be defined
Deleted: in order to manage file size
Comment [KH4]: Bring this heading up a level.
Deleted: can be found
Deleted: groupings
Deleted: (described above) on
Deleted: which
Deleted: We begin our review with
Deleted: . In this
Deleted: we define
Deleted: new
Deleted: (
Deleted:)
Deleted: This new
Deleted: a new (
Deleted:)
Deleted: This means you deploy a
Deleted: under
Deleted: new (
Deleted:)
Deleted: (shared-infra_hosts)
Deleted: segment
Deleted: OpenStack-Ansible
Deleted: (described above) which
Deleted: Next, we review
Deleted: . Here, we
Deleted: new group
Deleted: In this case we identify the new
Deleted: as
Deleted: OpenStack-Ansible
Deleted: OpenStack-Ansible
Deleted: whether
Deleted: may

Customizing existing components

[Provide an introductory paragraph here.]

Deploy directly on hosts

To deploy a component directly on the host instead of within a container, set the `is_metal` property to `true` for the container group in the `container_skel` section in the appropriate file.

The use of `container_vars` and mapping from container groups to host groups is the same for a service deployed directly onto the host.

Note: The `cinder-volume` component is deployed directly on the host by default. See the `env.d/cinder.yml` file for this example.

Omit a service or component from the deployment

To omit a component from a deployment, you can use one of several options:

- Remove the `physical_skel` link between the container group and the host group by deleting the related file located in the `env.d/` directory.
- Do not run the playbook that installs the component. Unless you specify the component to run directly on a host by using the `is_metal` property, a container creates for this component.
- Adjust the affinity to 0 for the host group. Unless you specify the component to run directly on a host by using the `is_metal` property, a container creates for this component.

Deploy existing components on dedicated hosts

To deploy a shared-infra component to dedicated hosts, modify the files that specify the host groups and container groups for the component.

For example, to run Galera directly on dedicated hosts, you would perform the following steps:

1. Modify the `container_skel` section of the `env.d/galera.yml` file. For example:

```
container_skel:
```

Deleted: ing

Deleted: under

Deleted: also

Deleted: exist

Deleted: . The simplest way to do this is to delete

Deleted: which

Deleted: ,

Comment [KH5]: This doesn't sound correct. A container is created? Or, more likely, a container is not created?

Comment [KH6]: This link is broken.

Comment [KH7]: Again, this doesn't sound correct. A container is created? Or, more likely, a container is not created?

Deleted: ing

Comment [KH8]:
shared infrastructure component
or
`shared-infra` component

Deleted: on

Deleted: both

Deleted: specifying

Formatted: List Paragraph, Indent: Left: 0.31", Numbered + Level: 1 + Numbering Style: 1, 2, 3, … + Start at: 1 + Alignment: Left + Aligned at: 0.29" + Indent at: 0.54"

Deleted: segment

Deleted: might look like

```
  galera_container:
    belongs_to:
      - db_containers
    contains:
      - galera
    properties:
      log_directory: mysql_logs
      service_name: galera
      is_metal: true
```

Note: To deploy within containers on the dedicated hosts, omit the `is_metal: true` property.

2. Assign the `db_containers` container group (from the preceding step) to a host group by providing a `physical_skel` section for the host group in a new or existing file, such as `env.d/galera.yml`. For example:

```
physical_skel:
  db_containers:
    belongs_to:
      - all_containers
  db_hosts:
    belongs_to:
      - hosts
```

3. Define the host group (`db_hosts`) in a `conf.d/` file (such as `galera.yml`). For example:

```
db_hosts:
  db-host1:
    ip: 172.39.123.11
  db-host2:
    ip: 172.39.123.12
  db-host3:
    ip: 172.39.123.13
```

Note: Each of the custom group names in this example (`db_containers` and `db_hosts`) are arbitrary. Choose your own group names, but ensure that the references are consistent among all relevant files.

Deleted: If you want to

Deleted: se

Deleted: We include it here as a recipe for the more commonly requested layout.

Deleted: Since we define the new container group (`db_containers` above), we must assign that container group to a host group. To assign

Deleted: new

Deleted: new

Deleted: ,

Deleted: e

Deleted: new

Deleted: (

Deleted:)

Deleted: Lastly, d

Formatted: code_body Char, Font: 11 pt, Font color: Auto

Deleted: above

Deleted: were

Deleted: You can c

Deleted: be sure

Deleted: between

Appendix D: Security

Security is one of the top priorities within OpenStack-Ansible (OSA), and many security enhancements for OpenStack clouds are available in deployments by default. This appendix provides a detailed overview of the most important security enhancements.

Note: Every deployer has different security requirements based on business needs, regulatory requirements, or end-user demands. The OpenStack Security Guide has instructions and advice on how to operate and consume an OpenStack cloud by using the most secure methods.

Encrypted communication

Any OpenStack cloud has sensitive information transmitted between services, including user credentials, service credentials, or information about resources being created. Encrypting this traffic is critical in environments where the network cannot be trusted. (For more information about securing the network, see the :ref:`least-access-openstack-services` section.)

Many of the services deployed with OSA are encrypted by default or offer encryption as an option. The playbooks generate self-signed certificates by default, but deployers have the option to use their existing certificates, keys, and CA certificates.

To learn more about how to customize the deployment of encrypted communications, see Securing services with SSL certificates.

Host security hardening

OSA provides a comprehensive security hardening role that applies over 200 security configurations as recommended by the Security Technical Implementation Guide (STIG) provided by the Defense Information Systems Agency (DISA). These security configurations are widely used and are distributed in the public domain by the United States government.

Comment [KH1]: Because Appendix F has advanced security configuration information in it, we might consider referring users to Appendix F from this intro. Add a sentence here, something like: "For information about configuring security, see Appendix F."

Deleted: serves as

Deleted: improvements

Deleted: will have

Deleted: their

Deleted: official

Deleted: plenty of

Deleted: via

Deleted: will have

Deleted: . This information

Deleted: es

Deleted: may not

Deleted: Review

Formatted: Font: Not Italic

Deleted: below for more details on securing the network

Deleted: OpenStack-Ansible

Deleted: review

Deleted: the

Deleted: documentation section

Deleted: OpenStack-Ansible offers

Deleted: uses

Deleted: Government

Host security hardening is required by several compliance and regulatory programs, such as the Payment Card Industry Data Security Standard (PCI DSS) (Requirement 2.2).

By default, OSA automatically applies the security hardening role to all deployments. The role has been carefully designed to perform as follows:

- Apply nondisruptively to a production OpenStack environment
- Balance security with OpenStack performance and functionality
- Run as quickly as possible

For more information about configuring the role in OSA, see :ref:`security_hardening`.

Isolation

By default, OSA provides isolation between the containers that run the OpenStack infrastructure (control plane) services and also between the virtual machines that end users spawn within the deployment. This isolation is critical because it can prevent container or virtual machine breakouts, or at least reduce the damage that breakouts might cause.

The Linux Security Modules (LSM) framework allows administrators to set mandatory access controls (MAC) on a Linux system. MAC is different than discretionary access controls (DAC) because the kernel enforces strict policies that no user can bypass. Although any user might be able to change a DAC policy (such as `chown bob secret.txt`), only the `root` user can alter a MAC policy.

OSA currently uses AppArmor to provide MAC policies on infrastructure servers and hypervisors. The AppArmor configuration sets the access policies to prevent one container from accessing the data of another container. For virtual machines, `libvirtd` uses the sVirt extensions to ensure that one virtual machine cannot access the data or devices from another virtual machine.

These policies are applied and governed at the kernel level. Any process that violates a policy is denied access to the resource. All denials are logged in `auditd` and are available at `/var/log/audit/audit.log`.

Deleted: see

Formatted: Font: Not Italic

Deleted: OpenStack-Ansible

Deleted: by default, but this can be disabled via an Ansible variable

Deleted: -

Deleted: Refer to the documentation on

Deleted: for more information on the role in OpenStack-Ansible

Deleted: OpenStack-Ansible

Deleted: by default

Deleted: since

Deleted: they may

Deleted: This

Deleted: cannot be

Deleted: ed by any user

Deleted: a

Deleted: may

Deleted: they cannot

Deleted: This privilege is reserved for the `root` user.

Deleted: OpenStack-Ansible

Deleted: control plane

Deleted: as well as

Deleted: will be

Deleted: with

Least privilege

The principle of least privilege is used throughout OSA to limit the damage that could be caused if an attacker gains access to any credentials.

OSA configures unique username and password combinations for each service that interacts with RabbitMQ and Galera/MariaDB. Each service that connects to RabbitMQ uses a separate virtual host for publishing and consuming messages. The MariaDB users for each service are granted access only to the databases that they need to query.

Securing network access to OpenStack services

OpenStack clouds provide many services to end users that enable them to build instances, provision storage, and create networks. Each of these services exposes one or more service ports and API endpoints to the network.

However, some of the services within an OpenStack cloud are accessible to all end users, while others are accessible only to administrators or operators on a secured network.

- Services that *all end users* can access
  - These services include Compute (nova), Object Storage (swift), Networking (neutron), and Image (glance).
  - These services should be offered on a sufficiently restricted network that still allows all end users to access the services.
  - A firewall must be used to restrict access to the network.
- Services that *only administrators or operators* can access
  - These services include MariaDB, Memcached, RabbitMQ, and the admin API endpoint for the Identity (keystone) service.
  - These services *must* be offered on a highly restricted network that is available only to administrative users.
  - A firewall must be used to restrict access to the network.

Limiting access to these networks has several benefits:

- Allows for network monitoring and alerting

Deleted: OpenStack-Ansible

Deleted: OpenStack-Ansible

Deleted: talks to

Deleted: only

Deleted: database(s)

Deleted: offer

Deleted: which allow

Deleted: clouds

Deleted: should be

Deleted: exposed

Deleted: should only be

Deleted: available

Deleted: OpenStack services fit into one of two criteria:¶

Formatted: Font: Not Bold, Italic

Deleted: This includes

Deleted: such as

Formatted: Font: Not Bold, Italic

Deleted: This includes services such as

Deleted: memcached

Formatted: Font: Not Bold, Italic

- Prevents unauthorized network surveillance
- Reduces the chance of credential theft
- Reduces damage from unknown or unpatched service vulnerabilities

OSA deploys HAProxy back ends for each service and restricts access for highly sensitive services by making them available only on the management network. Deployers with external load balancers must ensure that the back ends are configured securely and that firewalls prevent traffic from crossing between networks.

For more information about recommended network policies for OpenStack clouds, see the API endpoint process isolation and policy section of the OpenStack Security Guide.

Deleted: OpenStack-Ansible

Deleted: backends

Deleted: backends

Deleted: details on

Deleted: refer to

Deleted: from

Appendix E: Container networking

OpenStack-Ansible (OSA) deploys Linux containers (LXC) and uses Linux bridging between the container interfaces and the host interfaces to ensure that all traffic from containers flows over multiple host interfaces. This appendix describes how the interfaces are connected and how traffic flows.

For more information about how the OpenStack Networking service (neutron) uses the interfaces for instance traffic, see the OpenStack Networking Guide.

Bonded network interfaces

In a typical production environment, physical network interfaces are combined in bonded pairs for better redundancy and throughput. Avoid using two ports on the same multiport network card for the same bonded interface, because a network card failure affects both of the physical network interfaces used by the bond.

Linux bridges

The combination of containers and flexible deployment options requires implementation of advanced Linux networking features, such as bridges and namespaces.

- Bridges provide layer 2 connectivity (similar to switches) among physical, logical, and virtual network interfaces within a host. After a bridge is created, the network interfaces are virtually plugged in to it.

  OSA uses bridges to connect physical and logical network interfaces on the host to virtual network interfaces within containers.

- Namespaces provide logically separate layer 3 environments (similar to routers) within a host. Namespaces use virtual interfaces to connect with other namespaces, including the host namespace. These interfaces, often called `veth` pairs, are virtually plugged in between namespaces, similar to patch cables connecting physical devices such as switches and routers.

  Each container has a namespace that connects to the host namespace with one or more `veth` pairs. Unless specified, the system generates random names for `veth` pairs.

Deleted: LXC machine

Deleted: linux

Comment [KH1]: I don't think you need to say what issue the architecture avoids. It is enough to just state how it works.

Deleted: flow

Deleted: This

Deleted: is to avoid traffic flowing through the default LXC bridge which is a single host interface (and therefore could become a bottleneck), and which is interfered with by iptables.¶

Deleted: intends to

Deleted: details

Deleted: please

Deleted: A

Deleted: uses multiple

Deleted: a

Deleted: pair

Deleted: We recommend avoiding the use of

Deleted: -

Deleted: . This is

Deleted: require

Formatted: List Paragraph, Indent: Left: 0", Bulleted + Level: 1 + Aligned at: 0.25" + Indent at: 0.5"

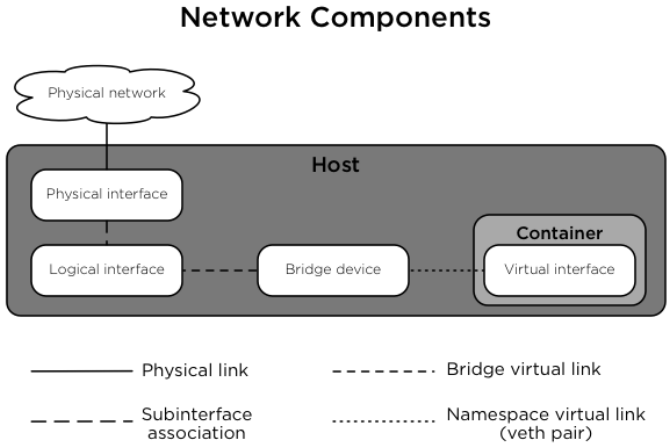
Deleted: creating

Deleted: OpenStack-Ansible

Formatted: List Paragraph, Indent: Left: 0", Bulleted + Level: 1 + Aligned at: 0.25" + Indent at: 0.5"

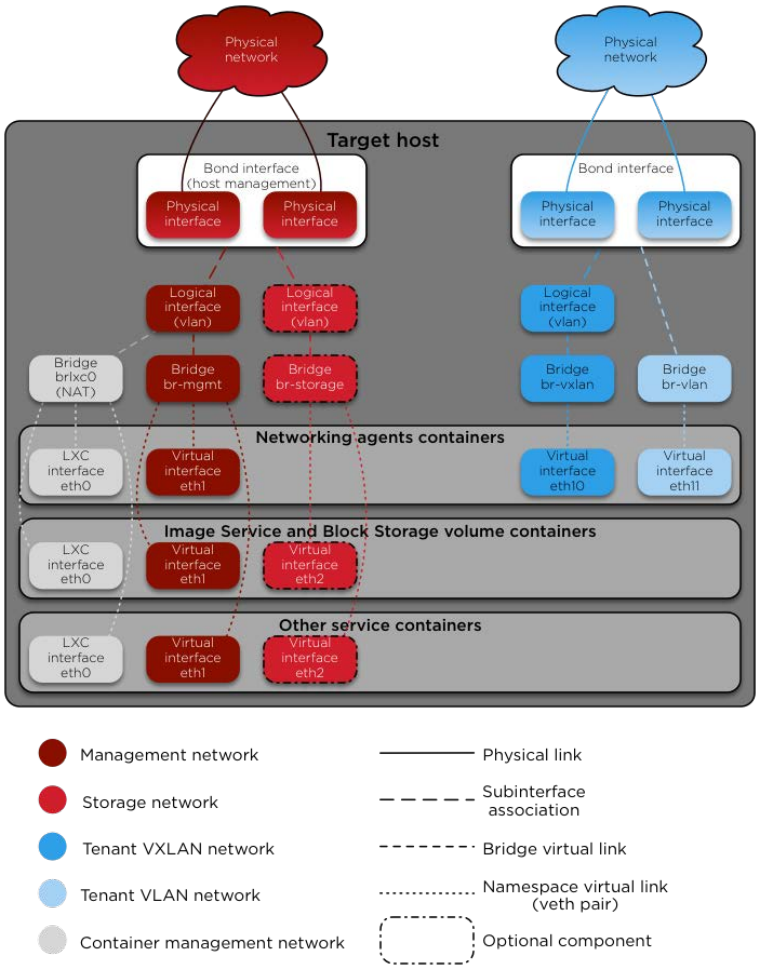
The following image demonstrates how the container network interfaces are connected to the host's bridges and physical network interfaces:

Deleted: to the host's



Network diagrams

The following digram shows how all of the interfaces and bridges interconnect to provide network connectivity to the OpenStack deployment:



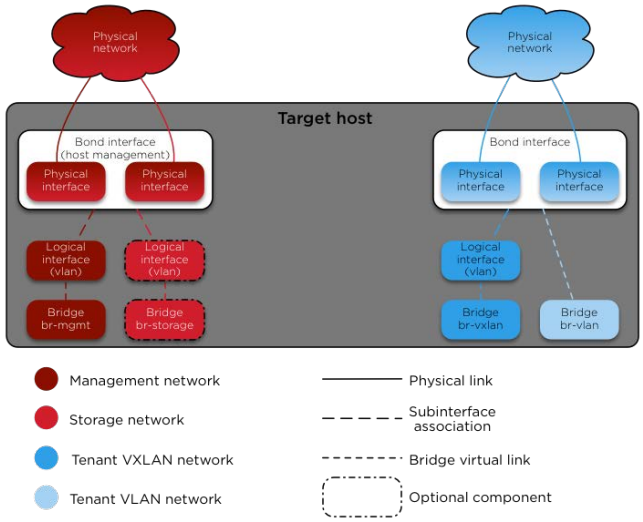
Comment [KH2]: Rather than having one general heading for all four diagrams, consider creating a separate descriptive heading for each one. This one could be "Overall network connectivity" (or something like that).

Deleted: image

OSA deploys the Compute service on the physical host rather than in a container. The following diagram shows how to use bridges for network connectivity:

Deleted: OpenStack-Ansible

Deleted: image



The following diagram shows how the Networking service (neutron) agents work with the `br-vlan` and `br-vxlan` bridges. Neutron is configured to use a DHCP agent, an L3 agent, and a Linux Bridge agent within a `networking-agents` container. The diagram shows how DHCP agents provide information (IP addresses and DNS servers) to the instances, and how routing works on the image.

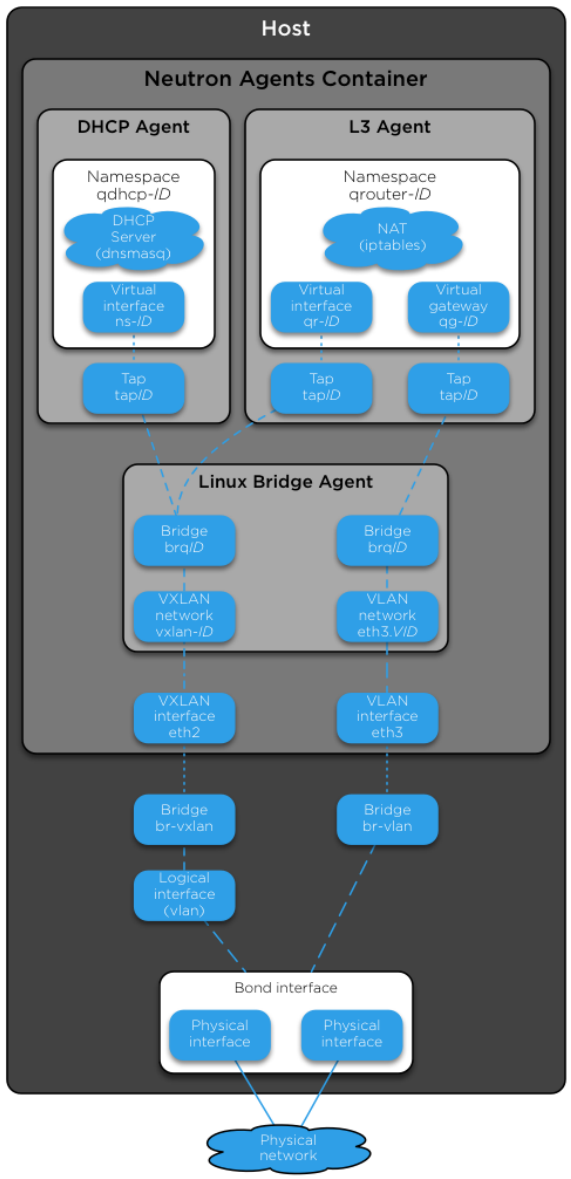
Deleted: image

Deleted: bridges

Deleted: image

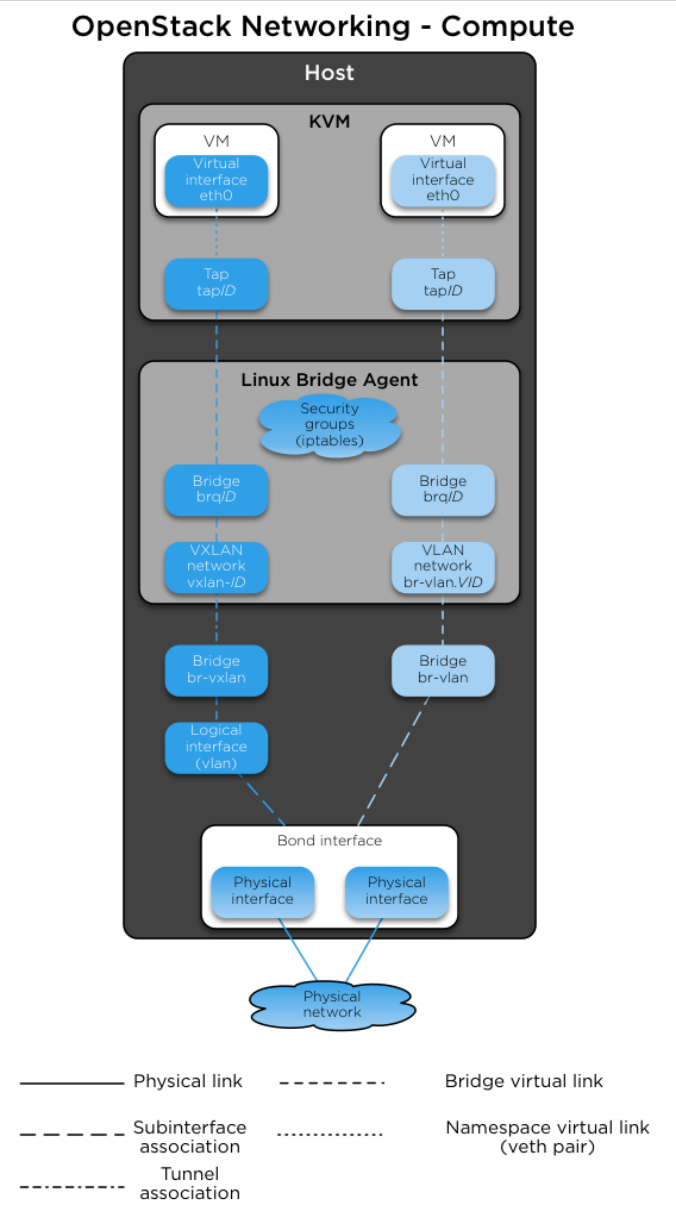
Deleted: :

OpenStack Networking - Neutron Agents



The following diagram shows how virtual machines connect to the `br-vlan` and `br-vxlan` bridges and send traffic to the network outside the host:

Deleted: image



Appendix F: Advanced configuration

[I suggest adding an intro paragraph that will appear on the main appendix page, following the TOC (here). It should explain what advanced configuration is, whether it's required or optional, and when someone should do it. Users coming to this appendix should immediately get a sense of whether it applies to them and their situation or not.]

Overriding OpenStack configuration defaults

OpenStack has many configuration options available in `.conf` files (in a standard `INI` file format), policy files (in a standard `JSON` format), and `YAML` files.

Note: `YAML` files are only in the ceilometer project at this time.

OpenStack-Ansible (OSA) enables you to reference any options in the OpenStack Configuration Reference through the use of a simple set of configuration entries in the `/etc/openstack_deploy/user_variables.yml` file. This section describes how to use the configuration entries in the `/etc/openstack_deploy/user_variables.yml` file to override default configuration settings.

For more information, see the Setting overrides in configuration files section in the developer documentation.

Overriding `.conf` files

Most often, overrides are implemented for the `<service>.conf` files (for example, `nova.conf`). These files use a standard `INI` file format.

For example, you might want to add the following parameters to the `nova.conf` file:

```
[DEFAULT]
remove_unused_original_minimum_age_seconds = 43200

[libvirt]
cpu_mode = host-model
disk_cachemodes = file=directsync,block=none

[database]
idle_timeout = 300
max_pool_size = 10
```

Deleted: configuration files which are in the form of

Deleted: also

Comment [KH1]: I am a little confused by this. Most of the basic configuration files that have been mentioned in the guide so far are **.yml** files. In fact, the main configuration file mentioned in this section is a **.yml** file (`user_variables.yml`). Isn't **.yml** a file extension for YAML files? If so, then this statement seems a little confusing, because obviously YAML files aren't only in the ceilometer project (unless I am reading it wrong).

Deleted: provides the facility to include

Deleted: to

Deleted: ¶
¶

Deleted: provides guidance for

Deleted: make

Deleted: of this facility. Further guidance is available in the developer documentation in the section titled

Comment [KH2]: Do not apply formatting to parts of headings. Use just normal format for **.conf**.

Deleted: The most common use-case for implementing

Deleted: if

To do this, you use the following configuration entry in the `/etc/openstack_deploy/user_variables.yml` file:

```
nova_nova_conf_overrides:
  DEFAULT:
    remove_unused_original_minimum_age_seconds: 43200
  libvirt:
    cpu_mode: host-model
    disk_cachemodes: file=directsync,block=none
  database:
    idle_timeout: 300
    max_pool_size: 10
```

Note: The general format for the variable names used for overrides is `<service>_<filename>_<file extension>_overrides`. For example, the variable name used in these examples to add parameters to the `nova.conf` file is `nova_nova_conf_overrides`.

You can also apply overrides on a per-host basis with the following configuration in the `/etc/openstack_deploy/openstack_user_config.yml` file:

```
compute_hosts:
  900089-compute001:
    ip: 192.0.2.10
    host_vars:
      nova_nova_conf_overrides:
        DEFAULT:
          remove_unused_original_minimum_age_seconds: 43200
        libvirt:
          cpu_mode: host-model
          disk_cachemodes: file=directsync,block=none
        database:
          idle_timeout: 300
          max_pool_size: 10
```

Use this method for any files with the `INI` format for in OpenStack projects deployed in OSA.

Overriding `.json` files

To implement access controls that are different from the ones in a standard OpenStack environment, you can adjust the default policies applied by services. Policy files are in a `JSON` format.

Deleted: This is accomplished through the

Deleted: of

Comment [KH3]: Make this a note (or a tip), and place it here. Having the information here (rather than at the end of the section) helps users understand the examples.

Deleted: O

Deleted: may also be applied

Deleted: file

Deleted: all

Deleted: OpenStack-Ansible

Deleted: To assist you in finding the appropriate variable name to use for overrides, the general format for the variable name is:`<service>_<filename>_<file extension>_overrides`.

Comment [KH4]: Do not apply formatting to parts of headings. Use just normal format for **.json**.

Deleted: Y

Deleted: in order to implement access controls which are different to a standard OpenStack environment

For example, you might want to add the following policy in the `policy.json` file for the Identity service (keystone):

```
{
    "identity:foo": "rule:admin_required",
    "identity:bar": "rule:admin_required"
}
```

To do this, you use the following configuration entry in the `/etc/openstack_deploy/user_variables.yml` file:

```
keystone_policy_overrides:
  identity:foo: "rule:admin_required"
  identity:bar: "rule:admin_required"
```

Note: The general format for the variable names used for overrides is `<service>_policy_overrides`. For example, the variable name used in this example to add a policy to the Identity service (keystone) `policy.json` file is `keystone_policy_overrides`.

Use this method for any files with the `JSON` format in OpenStack projects deployed in OSA.

Overriding YAML files

You can override `.yml` file values by supplying replacement YAML content.

Note: All default YAML file content is completely overwritten by the overrides, so the entire YAML source (both the existing content and your changes) must be provided.

For example, you might want to define a meter exclusion for all hardware items in the default content of the `pipeline.yml` file for the Telemetry service (ceilometer):

```
sources:
    - name: meter_source
    interval: 600
    meters:
        - "!hardware.*"
    sinks:
        - meter_sink
    - name: foo_source
    value: foo
```

Deleted: can

Deleted: keystone's

Deleted: Accomplish this through the

Deleted: of

Comment [KH5]: Make this a note (or a tip), and place it here. Having the information here (rather than at the end of the section) helps users understand the examples.

Deleted: all

Deleted: OpenStack-Ansible

Deleted: with `JSON` file formats

Deleted: To assist you in finding the appropriate variable name to use for overrides, the general format for the variable name is`<service>_policy_overrides`.

Comment [KH6]: Change this to **.yml** or **.yaml**, whichever is correct. (The intro paragraph uses .yml, but there is some confusion as to which extension is correct. Are they both correct? Or just one of them? If both, then retain the use of YAML in the heading. If just one, then be consistent.)

Deleted: will be

Deleted: provided

Deleted: can

Deleted: ceilometer's

To do this, you use the following configuration entry in the `/etc/openstack_deploy/user_variables.yml` file:

```
ceilometer_pipeline_yaml_overrides:
  sources:
      - name: meter_source
      interval: 600
      meters:
          - "!hardware.*"
      sinks:
          - meter_sink
      - name: source_foo
      value: foo
```

Note: The general format for the variable names used for overrides is `<service>_<filename>_<file extension>_overrides`. For example, the variable name used in this example to define a meter exclusion in the `pipeline.yml` file for the Telemetry service (ceilometer) is `ceilometer_pipeline_yaml_overrides`.

Currently available overrides

The following override variables are available.

Galera

- galera_client_my_cnf_overrides
- galera_my_cnf_overrides
- galera_cluster_cnf_overrides
- galera_debian_cnf_overrides

Telemetry service (ceilometer)

- ceilometer_policy_overrides
- ceilometer_ceilometer_conf_overrides
- ceilometer_api_paste_ini_overrides
- ceilometer_event_definitions_yaml_overrides
- ceilometer_event_pipeline_yaml_overrides
- ceilometer_pipeline_yaml_overrides

Deleted: You can accomplish this through the

Deleted: of

Comment [KH7]: The paragraph that precedes the example says `pipeline.yml`, not `pipeline.yaml`.

Comment [KH8]: Use whatever the correct extension is here.

Deleted: To assist you in finding the appropriate variable name to use for overrides, the general format for the variable name is`<service>_<filename>_<file extension>_overrides`.

Deleted: is a list of

Deleted: :

Deleted: :

Deleted: C

Deleted: :

Block Storage service (cinder)

Deleted: C

Deleted: :

- cinder_policy_overrides
- cinder_rootwrap_conf_overrides
- cinder_api_paste_ini_overrides
- cinder_cinder_conf_overrides

Image service (glance)

Deleted: G

Deleted: :

- glance_glance_api_paste_ini_overrides
- glance_glance_api_conf_overrides
- glance_glance_cache_conf_overrides
- glance_glance_manage_conf_overrides
- glance_glance_registry_paste_ini_overrides
- glance_glance_registry_conf_overrides
- glance_glance_scrubber_conf_overrides
- glance_glance_scheme_json_overrides
- glance_policy_overrides

Orchestration service (heat)

Deleted: H

Deleted: :

- heat_heat_conf_overrides
- heat_api_paste_ini_overrides
- heat_default_yaml_overrides
- heat_aws_cloudwatch_alarm_yaml_overrides
- heat_aws_rds_dbinstance_yaml_overrides
- heat_policy_overrides

Identity service (keystone)

Deleted: K

Deleted: :

- keystone_keystone_conf_overrides
- keystone_keystone_default_conf_overrides
- keystone_keystone_paste_ini_overrides
- keystone_policy_overrides

Networking service (neutron)

Deleted: N

Deleted: :

- neutron_neutron_conf_overrides

- neutron_ml2_conf_ini_overrides
- neutron_dhcp_agent_ini_overrides
- neutron_api_paste_ini_overrides
- neutron_rootwrap_conf_overrides
- neutron_policy_overrides
- neutron_dnsmasq_neutron_conf_overrides
- neutron_l3_agent_ini_overrides
- neutron_metadata_agent_ini_overrides
- neutron_metering_agent_ini_overrides

Compute service (nova)

Deleted: N

Deleted: :

- nova_nova_conf_overrides
- nova_rootwrap_conf_overrides
- nova_api_paste_ini_overrides
- nova_policy_overrides

Object Storage service (swift)

Deleted: S

Deleted: :

- swift_swift_conf_overrides
- swift_swift_dispersion_conf_overrides
- swift_proxy_server_conf_overrides
- swift_account_server_conf_overrides
- swift_account_server_replicator_conf_overrides
- swift_container_server_conf_overrides
- swift_container_server_replicator_conf_overrides
- swift_object_server_conf_overrides
- swift_object_server_replicator_conf_overrides

Tempest

Comment [KH9]: What is Tempest? Should we qualify it here with something? Something like:

Tempest (testing project)

- tempest_tempest_conf_overrides

pip

Deleted: :

Deleted: :

- pip_global_conf_overrides

Security hardening

OpenStack-Ansible (OSA) automatically applies host security hardening configurations by using the openstack-ansible-security role. The role uses a version of the Security Technical Implementation Guide (STIG) that has been adapted for Ubuntu 14.04 and OpenStack.

The role is applicable to physical hosts within an OSA deployment that are operating as any type of node, infrastructure or compute. By default, the role is enabled. You can disable it by changing the value of the `apply_security_hardening` variable in the `user_variables.yml` file to `false`:

```
apply_security_hardening: false
```

You can apply security hardening configurations to an existing environment or audit an environment by using a playbook supplied with OSA:

```
# Apply security hardening configurations
openstack-ansible security-hardening.yml

# Perform a quick audit by using Ansible's check mode
openstack-ansible --check security-hardening.yml
```

For more information about the security configurations, see the OSA host security hardening documentation.

Comment [KH10]: What about Ubuntu 16.04?

Deleted: OpenStack-Ansible

Deleted: a

Deleted: with

Deleted: ¶
When the variable is set to `true`, the `setup-hosts.yml` playbook applies the role during deployments.¶

Deleted: OpenStack-Ansible

Deleted: ¶
```
# Apply security hardening configurations¶
openstack-ansible security-hardening.yml¶
```

Deleted: Refer to

Comment [KH11]: The two references that were in this paragraph pointed to the same doc, but to different versions of it. My edit suggests pointing to it just once. If you want to refer to it twice, ensure that the links are correct.

Deleted: openstack-ansible-security

Deleted: for more details on the security configurations. Review theConfiguration section of the openstack-ansible-security documentation to find out how to fine-tune certain security configurations.

Securing services with SSL certificates

The OpenStack Security Guide recommends providing secure communication between various services in an OpenStack deployment. The OpenStack-Ansible (OSA) project currently offers the ability to configure SSL certificates for secure communication with the following services:

- HAProxy
- Dashboard (horizon)
- Identity (keystone)
- RabbitMQ

For each service, you can either use self-signed certificates that are generated during the deployment process or provide SSL certificates, keys, and CA certificates from your own trusted certificate authority. Highly secured environments use trusted, user-provided certificates for as many services as possible.

Note: Perform all SSL certificate configuration in the `/etc/openstack_deploy/user_variables.yml` file and not in the playbook roles themselves.

Self-signed certificates

Self-signed certificates enable you to start quickly and encrypt data in transit. However, they do not provide a high level of trust for highly secure environments.

By default, self-signed certificates are used in (OSA). When self-signed certificates are used, you must disable certificate verification by using the following user variables, depending on your configuration. Add these variables in the `/etc/openstack_deploy/user_variables.yml` file.

```
keystone_service_adminuri_insecure: true
keystone_service_internaluri_insecure: true
```

Deleted: ¶

Deleted: H

Deleted: K

Deleted: have the option to

Deleted: ,

Deleted: Conduct

Deleted: ensure you are able

Deleted: you are able to

Deleted: The use of

Deleted: is currently the default

Deleted: OpenStack-Ansible

Deleted: being

Deleted: must be disabled

Setting subject data for self-signed certificates

Change the subject data of any self-signed certificate by using configuration variables. The configuration variable for each service is formatted as `<servicename>_ssl_self_signed_subject`. For example, to change the SSL certificate subject data for HAProxy, adjust the `/etc/openstack_deploy/user_variables.yml` file as follows:

```
haproxy_ssl_self_signed_subject: "/C=US/ST=Texas/L=San
Antonio/O=IT/CN=haproxy.example.com"
```

For more information about the available fields in the certificate subject, see the OpenSSL documentation for the req subcommand.

Generating and regenerating self-signed certificates

Self-signed certificates are generated for each service during the first run of the playbook.

To generate a new self-signed certificate for a service, you must set the `<servicename>_ssl_self_signed_regen` variable to `true` in one of the following ways:

- To force a self-signed certificate to regenerate, pass the variable to `openstack-ansible` on the command line. For example:

  ```
  # openstack-ansible -e "horizon_ssl_self_signed_regen=true" os-horizon-
  install.yml
  ```

- To force a self-signed certificate to regenerate with every playbook run, set the appropriate regeneration option to `true` in the `/etc/openstack_deploy/user_variables.yml` file. For example, if you have already run the `os-horizon` playbook but you want to regenerate the self-signed certificate, set the `horizon_ssl_self_signed_regen` variable to `true`:

  ```
  horizon_ssl_self_signed_regen: true
  ```

Note: Regenerating self-signed certificates replaces the existing certificates whether they are self-signed or user-provided.

Deleted: subject data

Deleted: To

Deleted: refer to

Deleted: 's

Deleted: on

Deleted: Generate s

Deleted: Note: Subsequent runs of the playbook do not generate new SSL certificates unless you

Deleted: set

Deleted: to

Deleted: .

Comment [KH12]: Does this method regenerate the certificate only once? Or does it set it to regenerate on every playbook run, like the following method? If just once, then revise this to say, "To force a self-signed certificate to regenerate once, ..."

If both methods do the same thing, then revise the intro and bullet text as follows:

To generate a new self-signed certificate for a service every time a playbook runs, you must set the `<servicename>_ssl_self_signed_regen` variable to `true` in one of the following ways:

- Pass the variable to ...
- Set the appropriate regeneration option to `true` in the ...

Formatted: Bulleted + Level: 1 + Aligned at: 0" + Indent at: 0.25"

Deleted: you can

Formatted: Bulleted + Level: 1 + Aligned at: 0" + Indent at: 0.25"

Deleted: ,

Deleted: in `/etc/openstack_deploy/user_variables.yml`

User-provided certificates

For added trust in highly secure environments, you can provide your own SSL certificates, keys, and CA certificates. Acquiring certificates from a trusted certificate authority is outside the scope of this document, but the Certificate Management section of the Linux Documentation Project explains how to create your own certificate authority and sign certificates.

Use the following process to deploy user-provided SSL certificates in OSA:

1. Copy your SSL certificate, key, and CA certificate files to the deployment host.
2. Specify the path to your SSL certificate, key, and CA certificate in the `/etc/openstack_deploy/user_variables.yml` file.
3. Run the playbook for that service.

For example, to deploy user-provided certificates for RabbitMQ, copy the certificates to the deployment host, edit the `/etc/openstack_deploy/user_variables.yml` file, and set the following three variables:

```
rabbitmq_user_ssl_cert:    /tmp/example.com.crt
rabbitmq_user_ssl_key:     /tmp/example.com.key
rabbitmq_user_ssl_ca_cert: /tmp/ExampleCA.crt
```

Then, run the playbook to apply the certificates:

```
# openstack-ansible rabbitmq-install.yml
```

The playbook deploys your user-provided SSL certificate, key, and CA certificate to each RabbitMQ container.

The process is identical for the other services. Replace `rabbitmq` in the preceding configuration variables with `horizon`, `haproxy`, or `keystone`, and then run the playbook for that service to deploy user-provided certificates to those services.

Deleted: Y

Deleted: for added trust in highly secure environments

Comment [KH13]: Or would it be "OpenStack"?

Deleted: Deploying

Deleted: is a three step process

Comment [KH14]: Copy them just anywhere on the host? Does it matter?

Deleted: Run

Deleted: to

Deleted: shown above

Deleted:

Affinity

When OpenStack-Ansible (OSA) generates its dynamic inventory, the *affinity* setting determines how many containers of a similar type are deployed on a single physical host.

Using `shared-infra_hosts` as an example, consider this `openstack_user_config.yml` configuration:

```
shared-infra_hosts:
  infra1:
    ip: 172.29.236.101
  infra2:
    ip: 172.29.236.102
  infra3:
    ip: 172.29.236.103
```

Three hosts are assigned to the `shared-infra_hosts` group. OSA ensures that each host runs a single database container, a single Memcached container, and a single RabbitMQ container. Each host has an affinity of 1 by default, which means that each host runs one of each container type.

If you are deploying a stand-alone Object Storage (swift) environment, you can skip the deployment of RabbitMQ. If you use this configuration, your `openstack_user_config.yml` file would look as follows:

```
shared-infra_hosts:
  infra1:
    affinity:
      rabbit_mq_container: 0
    ip: 172.29.236.101
  infra2:
    affinity:
      rabbit_mq_container: 0
    ip: 172.29.236.102
  infra3:
    affinity:
      rabbit_mq_container: 0
    ip: 172.29.236.103
```

This configuration deploys a Memcached container and a database container on each host, but no RabbitMQ containers.

Deleted: 's dynamic

Deleted:

Deleted: generation has a concept called affinity. This

Deleted: to

Formatted: code_body Char, Font: (Default) Times New Roman, Font color: Auto

Formatted: code_body Char, Font: (Default) Times New Roman, Font color: Auto

Deleted: ,

Deleted: OpenStack-Ansible

Deleted: memcached

Deleted: and that means

Deleted: will

Deleted: Y

Deleted: altogether. This is helpful when deploying a standalone swift environment.

Deleted: need

Deleted: like this

Deleted: The

Deleted: above

Deleted: memcached

Deleted: without

Deleted: the

Advanced service configuration

OpenStack-Ansible (OSA) has many options that you can use for the advanced configuration of services. Each role's documentation provides information about the available options.

The following options are optional.

Deleted: which can be used

Deleted: more insight into

Deleted: available and what they are used for

Deleted: not necessary to set - they are entirely

Infrastructure service roles

- galera_server
- haproxy_server
- memcached_server
- rabbitmq_server
- repo_build
- repo_server
- rsyslog_server

OpenStack service roles

- os_aodh
- os_ceilometer
- os_cinder
- os_glance
- os_gnocchi
- os_heat
- os_horizon
- os_ironic
- os_keystone
- os_magnum
- os_neutron
- os_nova
- os_rally
- os_sahara
- os_swift

- os_tempest

Other roles

- ansible-plugins
- apt_package_pinning
- ceph_client
- galera_client
- lxc_container_create
- lxc_hosts
- pip_install
- openstack_openrc
- openstack_hosts
- rsyslog_client

Appendix G: Additional resources

Ansible resources:

- Ansible Documentation
- Ansible Best Practices
- Ansible Configuration

OpenStack resources:

- OpenStack Documentation
- OpenStack SDK, CLI, and API Documentation
- OpenStack API Guide
- OpenStack Project Developer Documentation

Deleted: The following

Deleted: are useful to reference

Deleted: The following

Deleted: are useful to reference