OpenStack-Ansible Deploy Guide

Release 32.1.0.dev42

OpenStack-Ansible Contributors

Jan 28, 2026

CONTENTS

This guide provides instructions for deployers to use to perform an OpenStack-Ansible installation in either a test environment or a production environment.

Note

If you want to do a quick proof of concept of OpenStack, read the All-In-One quickstart Guide instead of this document. This document is a walkthrough of a deployment using OpenStack-Ansible, with all of its configurability.

Contents:

CHAPTER

ONE

OVERVIEW

Note

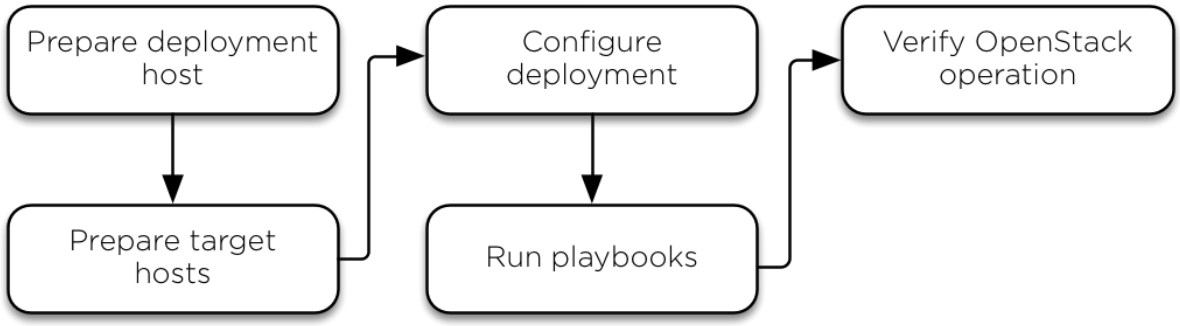
For essential background reading to help understand the service and storage architecture, please read the OpenStack-Ansible Architecture section of its reference guide. If youd like to understand when OpenStack-Ansible would be a good fit for your organisation, please read About OpenStack-Ansible.

This guide refers to the following types of hosts:

- *Deployment host*, which runs the Ansible playbooks
- *Target hosts*, where Ansible installs OpenStack services and infrastructure components

1.1 Installation workflow

The following diagram shows the general workflow of an OpenStack-Ansible installation.



1.2 Installation requirements and recommendations

1.2.1 Software requirements

Ensure that all hosts within the OpenStack-Ansible (OSA) environment meet the following minimum requirements:

Operating Systems:

Configuration:

- Secure Shell (SSH) client and server that support public key authentication
- Python 3.11.*x* or 3.12.*x*

- en_US.UTF-8 as the locale

1.2.2 CPU recommendations

- Compute hosts should have multicore processors with hardware-assisted virtualization extensions. These extensions provide a significant performance boost and improve security in virtualized environments.
- Infrastructure (control plane) hosts should have multicore processors for best performance. Some services, such as MariaDB, benefit from additional CPU cores and other technologies, such as Hyper-threading.

1.2.3 Storage/disk recommendations

Different hosts have different disk space requirements based on the services running on each host:

Deployment hosts

A minimum of `10 GB` of disk space is sufficient for holding the OpenStack-Ansible repository content and additional required software.

Compute hosts

Disk space requirements depend on the total number of instances running on each host and the amount of disk space allocated to each instance.

Tip

Consider disks that provide higher I/O throughput with lower latency, such as SSD or NVMe drives in a RAID array.

Storage hosts

Hosts running the Block Storage (cinder) service often consume the most disk space in OpenStack environments.

Tip

As with Compute hosts, choose disks that provide the highest I/O throughput with the lowest latency.

OpenStack-Ansible is able to deploy Cinder with a series of different backends and uses Logical Volume Manager (LVM), by default. Hosts that provide Block Storage volumes with LVM are recommended to have a large disk space available allocated to a `cinder-volume` volume group, which OpenStack-Ansible can configure for use with Block Storage.

Infrastructure (control plane) hosts

The OpenStack control plane contains storage-intensive services, such as the Image service (glance), and MariaDB. These hosts must have a minimum of `100 GB` of disk space.

Each infrastructure (control plane) host runs services inside machine containers. The container file systems are deployed by default on the root file system of each control plane host. You have the option to de-

ploy those container file systems into logical volumes by creating a volume group calledălxc. OpenStack-Ansible creates a 5 GB logical volume for the file system of each container running on the host.

Tip

Other technologies leveraging copy-on-write can be used to reduce the disk space requirements on machine containers.

1.2.4 Network recommendations

Note

You can deploy an OpenStack environment with only one physical network interface. This works for small environments, but it can cause problems when your environment grows.

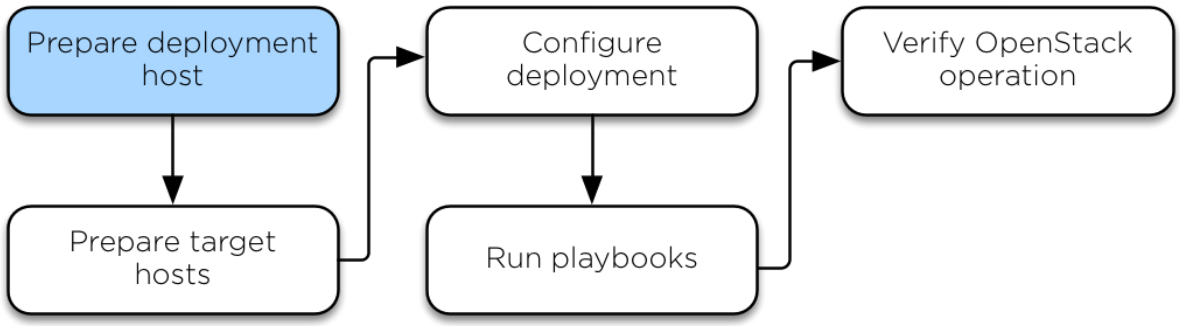
For the best performance, reliability, and scalability in a production environment, consider a network configuration that contains the following features:

- Bonded network interfaces, which increase performance, reliability, or both (depending on the bonding architecture).
- VLAN offloading, which increases performance by adding and removing VLAN tags in hardware, rather than in the servers main CPU.
- A high-speed Ethernet network can enhance storage performance when using the Block Storage service.
- Jumbo frames, which increase network performance by allowing more data to be sent in each packet.

CHAPTER

TWO

PREPARE THE DEPLOYMENT HOST



When you install OpenStack in a production environment, we recommend using a separate deployment host that contains Ansible and orchestrates the OpenStack-Ansible (OSA) installation on the target hosts. In a test environment, we recommend using one of the infrastructure target hosts as the deployment host.

To use a target host as a deployment host, follow the steps in Prepare the target hosts on the deployment host.

2.1 Configuring the operating system

2.1.1 Install the operating system

Install one of the following supported operating systems on the deployment hosts:

Configure at least one network interface to access the Internet or suitable local repositories.

2.1.2 Configure Ubuntu

Install additional software packages and configure Network Time Protocol (NTP). Before you begin, we recommend upgrading your system packages and kernel.

1. Update package source lists:

```
# apt update
```

2. Upgrade the system packages and kernel:

```
# apt dist-upgrade
```

3. Reboot the host.
4. Install additional software packages if they were not installed during the operating system installation:

```
# apt install build-essential git chrony openssh-server python3-dev sudo
```

5. Configure NTP to synchronize with a suitable time source.

2.1.3 Configure CentOS Stream / Rocky Linux

Install additional software packages and configure Network Time Protocol (NTP). Before you begin, we recommend upgrading your system packages and kernel.

1. Upgrade the system packages and kernel

```
# dnf upgrade
```

2. Disable SELinux. Edit `/etc/sysconfig/selinux`, make sure that `SELINUX=enforcing` is changed to `SELINUX=disabled`.

   For RHEL distributions starting from version 9 the recommended way to disable SELinux is via the boot loader using grubby:

```
# grubby --update-kernel ALL --args selinux=0
```

3. Reboot the host.
4. Install additional software packages if they were not installed during the operating system installation:

```
# dnf install git chrony openssh-server python3-devel sudo
# dnf group install "Development Tools"
```

5. Configure NTP to synchronize with a suitable time source and start chronyd:

```
# systemctl enable chronyd
# systemctl start chronyd
```

6. The `firewalld` service is enabled on CentOS Stream and Rocky Linux by default and its default ruleset prevents OpenStack components from communicating properly. Stop the `firewalld` service and mask it to prevent it from starting:

```
# systemctl stop firewalld
# systemctl mask firewalld
```

2.2 Configure SSH keys

Ansible uses SSH with public key authentication to connect the deployment host and target hosts. To reduce user interaction during Ansible operations, do not include passphrases with key pairs. However, if a passphrase is required, consider using the `ssh-agent` and `ssh-add` commands to temporarily store the passphrase before performing Ansible operations.

2.3 Configure the network

Ansible deployments fail if the deployment server cant use Secure Shell (SSH) to connect to the containers.

Configure the deployment host (where Ansible is executed) to be on the same layer 2 network as the network designated for container management. By default, this is the `br-mgmt` network. This configuration reduces the rate of failure caused by connectivity issues.

Select an IP address from the following example range to assign to the deployment host:

```
Container management: 172.29.236.0/22 (VLAN 10)
```

2.4 Install the source and dependencies

Install the source and dependencies for the deployment host.

Note

If you are installing with limited connectivity, please review Installing with limited connectivity before proceeding.

1. Clone the latest stable release of the OpenStack-Ansible Git repository in the `/opt/openstack-ansible` directory:

   ```
   # git clone -b master https://opendev.org/openstack/openstack-ansible /
   ↪opt/openstack-ansible
   ```

Warning

The value of `|latest_tag|` might not match the highest available tag. You can verify the latest stable release by checking the tag list here: OpenStack-Ansible Tags.

If opendev.org can not be accessed to run git clone, github.com can be used as an alternative repo:

```
# git clone -b master https://github.com/openstack/openstack-ansible.git /
↪opt/openstack-ansible
```

1. Change to the `/opt/openstack-ansible` directory, and run the Ansible bootstrap script:

   ```
   # scripts/bootstrap-ansible.sh
   ```

2.5 Configure Docker with Alpine

It is an alternative realization of deploy host configuration which includes usage of the Docker container as the deploy host.

This is also neither supported nor tested in CI, so you should use it at your own risk.

Before you begin, we recommend upgrading your Docker host system packages and kernel.

1. Prepare your OpenStack Ansible Dockerfile

```
FROM alpine
RUN apk add --no-cache bash build-base git python3-dev openssh-client␣
↪openssh-keygen sudo py3-virtualenv iptables libffi-dev openssl-dev␣
↪linux-headers coreutils curl
RUN git clone -b master https://git.openstack.org/openstack/
↪openstack-ansible /opt/openstack-ansible
WORKDIR /opt/openstack-ansible
RUN /opt/openstack-ansible/scripts/bootstrap-ansible.sh
ENTRYPOINT ["bash"]
```

2. Build and run your deploy host container

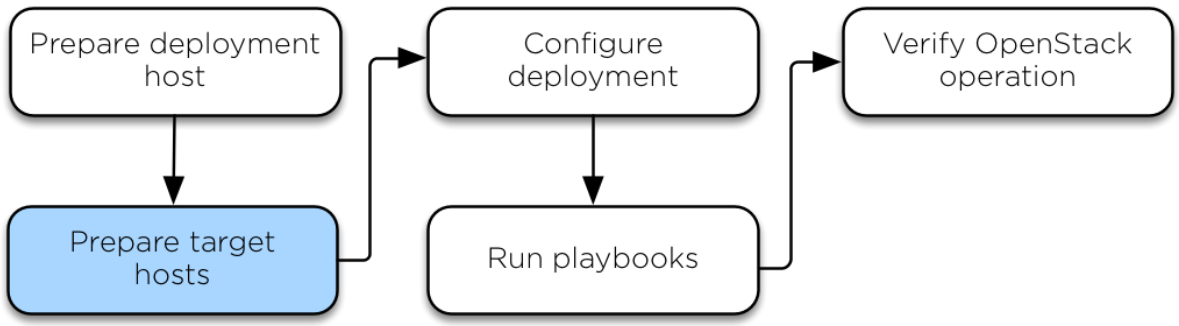
```
# docker build . -t openstack-ansible:master
# docker run -dit --name osa-deploy openstack-ansible:master
# docker exec -it osa-deploy bash
```

3. Configure NTP to synchronize with a suitable time source on the Docker host.

CHAPTER

THREE

PREPARE THE TARGET HOSTS



3.1 Configuring the operating system

This section describes the installation and configuration of operating systems for the target hosts, as well as deploying SSH keys and configuring storage.

3.1.1 Installing the operating system

Install one of the following supported operating systems on the target host:

Configure at least one network interface to access the Internet or suitable local repositories.

Some distributions add an extraneous entry in the `/etc/hosts` file that resolves the actual hostname to another loopback IP address such as `127.0.1.1`. You must comment out or remove this entry to prevent name resolution problems. **Do not remove the 127.0.0.1 entry.** This step is especially important for *metal* deployments.

Use short hostnames rather than fully-qualified domain names (FQDN) to prevent length limitation issues with LXC and SSH. For example, a suitable short hostname for a compute host might be: 12345-Compute001.

We recommend adding the Secure Shell (SSH) server packages to the installation on target hosts that do not have local (console) access.

Note

We also recommend setting your locale to *en_US.UTF-8*. Other locales might work, but they are not tested or supported.

3.1.2 Configure Debian

1. Update package source lists

```
# apt update
```

2. Upgrade the system packages and kernel:

```
# apt dist-upgrade
```

3. Install additional software packages:

```
# apt install bridge-utils debootstrap ifenslave ifenslave-2.6 \
  lsof lvm2 openssh-server sudo tcpdump vlan python3
```

4. Reboot the host to activate the changes and use the new kernel.

3.1.3 Configure Ubuntu

1. Update package source lists

```
# apt update
```

2. Upgrade the system packages and kernel:

```
# apt dist-upgrade
```

3. Install additional software packages:

```
# apt install bridge-utils debootstrap openssh-server \
  tcpdump vlan python3
```

4. Install the kernel extra package if you have one for your kernel version

```
# apt install linux-modules-extra-$(uname -r)
```

5. Reboot the host to activate the changes and use the new kernel.

3.1.4 Configure CentOS Stream / Rocky Linux

1. Upgrade the system packages and kernel:

```
# dnf upgrade
```

2. Disable SELinux. Edit `/etc/sysconfig/selinux`, make sure that `SELINUX=enforcing` is changed to `SELINUX=disabled`.

   # For RHEL distributions starting from version 9 the recommended way to disable SELinux is via the boot loader using grubby:

```
# grubby --update-kernel ALL --args selinux=0
```

Note

SELinux enabled is not currently supported in OpenStack-Ansible for CentOS/RHEL due to a lack of maintainers for the feature.

3. Disable firewalld for proper components communication:

```
# systemctl stop firewalld
# systemctl mask firewalld
```

4. Install additional software packages:

```
# dnf install iputils lsof openssh-server\
  sudo tcpdump python3
```

5. (Optional) Reduce the kernel log level by changing the printk value in your sysctls:

```
# echo "kernel.printk='4 1 7 4'" >> /etc/sysctl.conf
```

6. Reboot the host to activate the changes and use the new kernel.

3.2 Configure SSH keys

Ansible uses SSH to connect the deployment host and target hosts. You can either use `root` user or any other user that is allowed to escalate privileges through Ansible become (like adding user to sudoers). For more details, please reffer to the Running as non-root.

1. Copy the contents of the public key file on the deployment host to the `~/.ssh/authorized_keys` file on each target host.
2. Test public key authentication from the deployment host to each target host by using SSH to connect to the target host from the deployment host. If you can connect and get the shell without authenticating, it is working. SSH provides a shell without asking for a password.

For more information about how to generate an SSH key pair, as well as best practices, see GitHubs documentation about generating SSH keys.

3.3 Configuring the storage

Logical Volume Manager (LVM) enables a single device to be split into multiple logical volumes that appear as a physical storage device to the operating system. The Block Storage (cinder) service, and LXC containers that optionally run the OpenStack infrastructure, can optionally use LVM for their data storage.

Note

OpenStack-Ansible automatically configures LVM on the nodes, and overrides any existing LVM configuration. If you had a customized LVM configuration, edit the generated configuration file as needed.

1. To use the optional Block Storage (cinder) service, create an LVM volume group named `cinder-volumes` on the storage host. Specify a metadata size of 2048 when creating the physical volume. For example:

```
# pvcreate --metadatasize 2048 physical_volume_device_path
# vgcreate cinder-volumes physical_volume_device_path
```

2. Optionally, create an LVM volume group named `lxc` for container file systems and set `lxc_container_backing_store: lvm` in user_variables.yml if you want to use LXC with LVM. If the `lxc` volume group does not exist, containers are automatically installed on the file system under `/var/lib/lxc` by default.

3.4 Configuring the network

OpenStack-Ansible uses bridges to connect physical and logical network interfaces on the host to virtual network interfaces within containers. Target hosts need to be configured with the following network bridges:

Bridge name	Best configured on	With a static IP
br-mgmt	On every node	Always
br-storage	On every storage node	When component is deployed on metal
	On every compute node	Always
br-vxlan	On every network node	When component is deployed on metal
	On every compute node	Always
br-vlan	On every network node	Never
	On every compute node	Never

For a detailed reference of how the host and container networking is implemented, refer to OpenStack-Ansible Reference Architecture, section Container Networking.

For use case examples, refer to User Guides.

3.4.1 Host network bridges information

- LXC internal: `lxcbr0`

  The `lxcbr0` bridge is **required** for LXC, but OpenStack-Ansible configures it automatically. It provides external (typically Internet) connectivity to containers with dnsmasq (DHCP/DNS) + NAT.

  This bridge does not directly attach to any physical or logical interfaces on the host because iptables handles connectivity. It attaches to `eth0` in each container.

  The container network that the bridge attaches to is configurable in the `openstack_user_config.yml` file in the `provider_networks` dictionary.

- Container management: `br-mgmt`

  The `br-mgmt` bridge provides management of and communication between the infrastructure and OpenStack services.

  The bridge attaches to a physical or logical interface, typically a `bond0` VLAN subinterface. It also attaches to `eth1` in each container.

The container network interface that the bridge attaches to is configurable in the `openstack_user_config.yml` file.

- Storage: `br-storage`

  The `br-storage` bridge provides segregated access to Block Storage devices between OpenStack services and Block Storage devices.

  The bridge attaches to a physical or logical interface, typically a `bond0` VLAN subinterface. It also attaches to `eth2` in each associated container.

  The container network interface that the bridge attaches to is configurable in the `openstack_user_config.yml` file.

- OpenStack Networking tunnel: `br-vxlan`

  The `br-vxlan` interface is **required if** the environment is configured to allow projects to create virtual networks using VXLAN. It provides the interface for encapsulated virtual (VXLAN) tunnel network traffic.

  Note that `br-vxlan` is not required to be a bridge at all, a physical interface or a bond VLAN subinterface can be used directly and will be more efficient. The name `br-vxlan` is maintained here for consistency in the documentation and example configurations.

  The container network interface it attaches to is configurable in the `openstack_user_config.yml` file.

- OpenStack Networking provider: `br-vlan`

  Note

  The br-vlan bridge is no longer strictly necessary with right configuration you can use the physical interface directly (for example, in OVN). It remains in some setups mostly for consistency and to align naming conventions across documentation, but its use is optional.

  The `br-vlan` bridge is provides infrastructure for VLAN tagged or flat (no VLAN tag) networks.

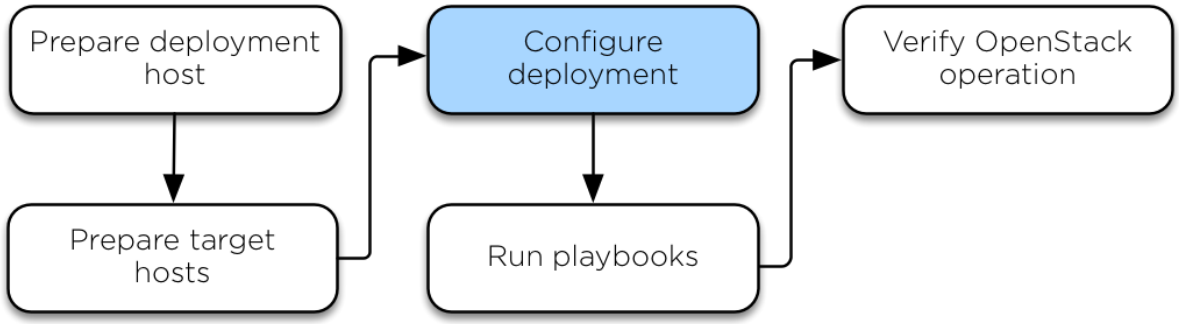
  The bridge attaches to a physical or logical interface, typically `bond1`. It is not assigned an IP address because it handles only layer 2 connectivity.

  The container network interface that the bridge attaches to is configurable in the `openstack_user_config.yml` file.

CHAPTER

FOUR

CONFIGURE THE DEPLOYMENT



Ansible references some files that contain mandatory and optional configuration directives. Before you can run the Ansible playbooks, modify these files to define the target environment. Configuration tasks include:

- Target host networking to define bridge interfaces and networks.
- A list of target hosts on which to install the software.
- Virtual and physical network relationships for OpenStack Networking (neutron).
- Passwords for all services.

4.1 Initial environment configuration

OpenStack-Ansible (OSA) depends on various files that are used to build an inventory for Ansible. Perform the following configuration on the deployment host.

1. Copy the contents of the `/opt/openstack-ansible/etc/openstack_deploy` directory to the `/etc/openstack_deploy` directory.

   ```
   # cp -a /opt/openstack-ansible/etc/openstack_deploy /etc/openstack_deploy
   ```

2. Change to the `/etc/openstack_deploy` directory.
3. Copy the `openstack_user_config.yml.example` file to `openstack_user_config.yml`

   ```
   # cp openstack_user_config.yml.example openstack_user_config.yml
   ```

4. Review the `openstack_user_config.yml` file and make changes to the deployment of your OpenStack environment.

> **Note**
>
> This file is heavily commented with details about the various options. See our User Guide and Reference Guide for more details.

5. Review the `user_variables.yml` file to configure global and role specific deployment options. The file contains some example variables and comments but you can get the full list of variables in each roles specific documentation.

> **Note**
>
> One important variable is the `install_method` which configures the installation method for the OpenStack services. The services can either be deployed from source (default) or from distribution packages. Source based deployments are closer to a vanilla OpenStack installation and allow for more tweaking and customizations. On the other hand, distro based deployments generally provide a package combination which has been verified by the distributions themselves. However, this means that updates are being released less often and with a potential delay. Moreover, this method offer fewer opportunities for deployment customizations and is supported only by selected services. The `install_method` variable is set during the initial deployment and you **must not** change it as OpenStack-Ansible is not able to convert itself from one installation method to the other. As such, its important to judge your needs against the pros and cons of each method before making a decision.

The configuration in the `openstack_user_config.yml` file defines which hosts run the containers and services deployed by OpenStack-Ansible. For example, hosts listed in the `shared-infra_hosts` section run containers for many of the shared services that your OpenStack environment requires. Some of these services include databases, Memcached, and RabbitMQ. Several other host types contain other types of containers, and all of these are listed in the `openstack_user_config.yml` file.

Some services, such as glance, heat, horizon and nova-infra, are not listed individually in the example file as they are contained in the os-infra hosts. You can specify image-hosts or dashboard-hosts if you want to scale out in a specific manner.

For examples, please see our User Guides

For details about how the inventory is generated, from the environment configuration and the variable precedence, see our Reference Guide under the inventory section.

4.2 Configure target hosts

Modify the `/etc/openstack_deploy/openstack_user_config.yml` file to configure the target hosts.

Do not assign the same IP address to different target hostnames. Unexpected results may occur. Each IP address and hostname must be a matching pair. To use the same host in multiple roles, for example infrastructure and networking, specify the same hostname and IP in each section.

Unless otherwise stated, replace `*_IP_ADDRESS` with the IP address of the br-mgmt container management bridge on each target host.

Note

If the SSH access to the host is via a different network than the br-mgmt interface, please, refer to the guide.

1. Configure a list containing at least three infrastructure target hosts in the `shared-infra_hosts` section:

```
shared-infra_hosts:
  infra01:
    ip: INFRA01_IP_ADDRESS
  infra02:
    ip: INFRA02_IP_ADDRESS
  infra03:
    ip: INFRA03_IP_ADDRESS
  infra04: ...
```

2. Configure a list of at least one keystone target host in the `identity_hosts` section:

```
identity_hosts:
  infra01:
    ip: INFRA01_IP_ADDRESS
  infra02: ...
```

3. Configure the appropriate set of hosts responsible for network-related roles in your deployment:

```
network-infra_hosts:
  infra01:
    ip: INFRA01_IP_ADDRESS
  infra02: ...

network-northd_hosts:
  infra01:
    ip: INFRA01_IP_ADDRESS
  infra02: ...
```

When deploying OpenStack with OVN, its essential to properly configure `network-gateway_hosts` depending on your network architecture. There are two typical scenarios:

Scenario 1: DVR with gateway on compute nodes:

```
network-gateway_hosts:
  compute01:
    ip: COMPUTE01_IP_ADDRESS
  compute02: ...
```

Scenario 2: standalone network nodes:

```
network-gateway_hosts:
  network01:
```

(continues on next page)

(continued from previous page)

```
    ip: NETWORK01_IP_ADDRESS
  network02: ...
```

4. Configure a list containing at least one compute target host in the `compute_hosts` section:

```
compute_hosts:
  compute01:
    ip: COMPUTE01_IP_ADDRESS
  compute02: ...
```

5. Configure a list containing at least one repository target host in the `repo-infra_hosts` section:

```
repo-infra_hosts:
  infra01:
    ip: INFRA01_IP_ADDRESS
  infra02:
    ip: INFRA02_IP_ADDRESS
  infra03:
    ip: INFRA03_IP_ADDRESS
  infra04: ...
```

   The repository typically resides on one or more infrastructure hosts.

6. Optionally configure storage host in the `storage_hosts` section:

```
storage_hosts:
  storage01:
    ip: STORAGE01_IP_ADDRESS
  storage02: ...
```

   Each storage host requires additional configuration to define the back end driver. The default configuration includes an optional storage host. To install without storage hosts, comment out the stanza beginning with the `storage_hosts:` line.

4.3 Installing additional services

To install additional services, the files in `etc/openstack_deploy/conf.d` provide examples showing the correct host groups to use. To add another service, add the host group, allocate hosts to it, and then execute the playbooks.

4.4 Advanced service configuration

OpenStack-Ansible has many options that you can use for the advanced configuration of services. Each roles documentation provides information about the available options.

Important

This step is essential to tailoring OpenStack-Ansible to your needs and is generally overlooked by new deployers. Have a look at each role documentation, user guides, and reference if you want a tailor made cloud.

4.4.1 Infrastructure service roles

- PKI
- galera_server
- haproxy_server
- memcached_server
- rabbitmq_server
- repo_server
- Zookeeper

4.4.2 OpenStack service roles

- os_adjutant
- os_aodh
- os_barbican
- os_ceilometer
- os_cinder
- os_cloudkitty
- os_designate
- os_glance
- os_gnocchi
- os_heat
- os_horizon
- os_ironic
- os_keystone
- os_magnum
- os_manila
- os_masakari
- os_mistral
- os_neutron
- os_nova
- os_octavia
- os_placement
- os_rally
- os_swift
- os_tacker
- os_tempest

- os_trove
- os_zun

4.4.3 Other roles

- apt_package_pinning
- ceph_client
- lxc_container_create
- lxc_hosts
- openstack_hosts
- openstack_openrc
- plugins
- python_venv_build
- systemd_service
- systemd_mount
- systemd_networkd
- unbound
- uWSGI

4.5 Configuring service credentials

Configure credentials for each service in the `/etc/openstack_deploy/user_secrets.yml` file. Consider using the Ansible Vault feature to increase security by encrypting any files that contain credentials. You can leverage osa_ops.encrypt_secrets collection to automate simplify the process of encryption and further management of secrets.

Adjust permissions on these files to restrict access by non-privileged users.

The `keystone_auth_admin_password` option configures the `admin` user password for both the OpenStack API and Dashboard access.

We recommend that you use the `pw-token-gen.py` script to generate random values for the variables in each file that contains service credentials:

```
# cd /opt/openstack-ansible
# ./scripts/pw-token-gen.py --file /etc/openstack_deploy/user_secrets.yml
```

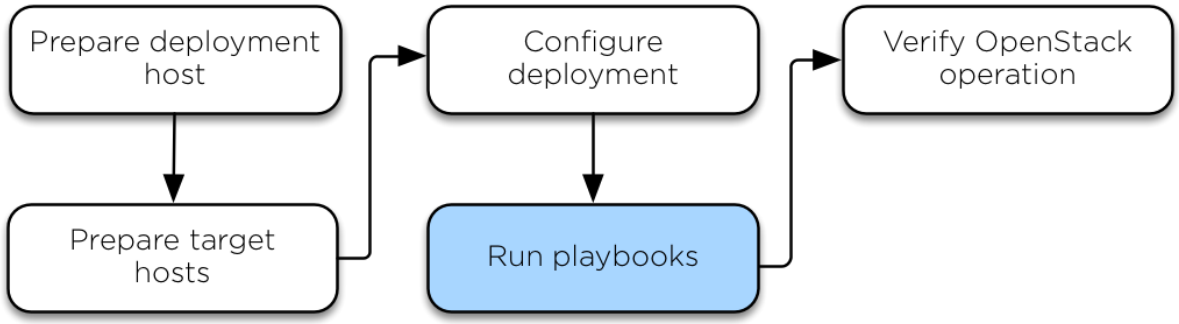
To regenerate existing passwords, add the `--regen` flag.

For information on how to rotate passwords, please refer to the Password Rotation documentation.

CHAPTER

FIVE

RUN PLAYBOOKS



The installation process requires running three main playbooks:

- The `openstack.osa.setup_hosts` Ansible foundation playbook prepares the target hosts for infrastructure and OpenStack services, builds and restarts containers on target hosts, and installs common components into containers on target hosts.
- The `openstack.osa.setup_infrastructure` Ansible infrastructure playbook installs infrastructure services: Memcached, the repository server, Galera and RabbitMQ.
- The `openstack.osa.setup_openstack` OpenStack playbook installs OpenStack services, including Identity (keystone), Image (glance), Block Storage (cinder), Compute (nova), Networking (neutron), etc.

5.1 Checking the integrity of the configuration files

Before running any playbook, check the integrity of the configuration files.

1. Ensure that all the files edited in the `/etc/openstack_deploy` directory are Ansible YAML compliant.
2. Check the integrity of your YAML files.

   Note

   To check your YAML syntax online, you can use the YAML Lint program.

3. Run the following command:

```
# openstack-ansible openstack.osa.setup_infrastructure --syntax-check
```

4. Recheck that all indentation is correct. This is important because the syntax of the configuration files can be correct while not being meaningful for OpenStack-Ansible.

5.2 Run the playbooks to install OpenStack

1. Run the host setup playbook:

```
# openstack-ansible openstack.osa.setup_hosts
```

Confirm satisfactory completion with zero items unreachable or failed:

```
PLAY RECAP␣
↪*********************************************************************
...
deployment_host                : ok=18   changed=11   unreachable=0   ␣
↪failed=0
```

2. Run the infrastructure setup playbook:

```
# openstack-ansible openstack.osa.setup_infrastructure
```

Confirm satisfactory completion with zero items unreachable or failed:

```
PLAY RECAP␣
↪*********************************************************************
...
deployment_host                : ok=27   changed=0    unreachable=0   ␣
↪failed=0
```

3. Run the following command to verify the database cluster:

Note

In order to run ad-hoc commands, you need to execute command from the location of `openstack-ansible` repository (ie */opt/openstack-ansible*) or explicitly load required environment variables for Ansible configuration through `source /usr/local/bin/openstack-ansible.rc`.

```
# ansible galera_container -m shell \
  -a "mariadb -h localhost -e 'show status like \"%wsrep_cluster_%\";'"
```

Example output:

```
node3_galera_container-3ea2cbd3 | success | rc=0 >>
Variable_name              Value
wsrep_cluster_conf_id      17
wsrep_cluster_size         3
wsrep_cluster_state_uuid   338b06b0-2948-11e4-9d06-bef42f6c52f1
wsrep_cluster_status       Primary
```

(continues on next page)

(continued from previous page)

```
node2_galera_container-49a47d25 | success | rc=0 >>
Variable_name            Value
wsrep_cluster_conf_id    17
wsrep_cluster_size       3
wsrep_cluster_state_uuid 338b06b0-2948-11e4-9d06-bef42f6c52f1
wsrep_cluster_status     Primary

node4_galera_container-76275635 | success | rc=0 >>
Variable_name            Value
wsrep_cluster_conf_id    17
wsrep_cluster_size       3
wsrep_cluster_state_uuid 338b06b0-2948-11e4-9d06-bef42f6c52f1
wsrep_cluster_status     Primary
```

The `wsrep_cluster_size` field indicates the number of nodes in the cluster and the `wsrep_cluster_status` field indicates primary.

4. Run the OpenStack setup playbook:

```
# openstack-ansible openstack.osa.setup_openstack
```

Confirm satisfactory completion with zero items unreachable or failed.

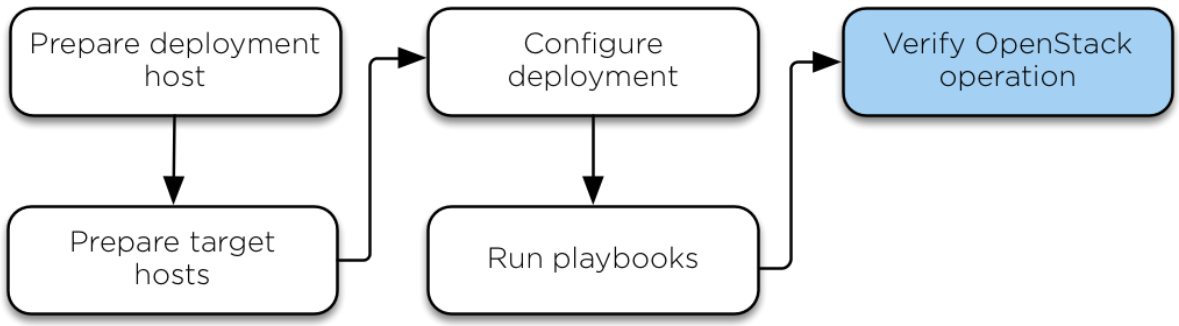
Note

You can also consider applying a hardening role for the deployment host to improve security. For more details, see the Apply ansible-hardening.

CHAPTER

SIX

VERIFYING OPENSTACK OPERATION



To verify basic operation of the OpenStack API and the Dashboard, perform the following tasks on an infrastructure host.

6.1 Verify the API

The utility container provides a CLI environment for additional configuration and testing.

1. Determine the name of the utility container:

```
# lxc-ls | grep utility
infra1_utility_container-161a4084
```

2. Access the utility container:

```
# lxc-attach -n infra1_utility_container-161a4084
```

3. Source the `admin` project credentials:

```
$ . ~/openrc
```

4. List your openstack users:

```
# openstack user list --os-cloud=default
+----------------------------------+--------------------+
| ID                               | Name               |
+----------------------------------+--------------------+
| 08fe5eeeae314d578bba0e47e7884f3a | alt_demo           |
| 0aa10040555e47c09a30d2240e474467 | dispersion         |
| 10d028f9e47b4d1c868410c977abc3df | glance             |
```

(continues on next page)

(continued from previous page)

```
| 249f9ad93c024f739a17ca30a96ff8ee | demo                |
| 39c07b47ee8a47bc9f9214dca4435461 | swift               |
| 3e88edbf46534173bc4fd8895fa4c364 | cinder              |
| 41bef7daf95a4e72af0986ec0583c5f4 | neutron             |
| 4f89276ee4304a3d825d07b5de0f4306 | admin               |
| 943a97a249894e72887aae9976ca8a5e | nova                |
| ab4f0be01dd04170965677e53833e3c3 | stack_domain_admin  |
| ac74be67a0564722b847f54357c10b29 | heat                |
| b6b1d5e76bc543cda645fa8e778dff01 | ceilometer          |
| dc001a09283a404191ff48eb41f0ffc4 | aodh                |
| e59e4379730b41209f036bbeac51b181 | keystone            |
+----------------------------------+---------------------+
```

6.2 Verifying the Dashboard (Horizon)

1. With a web browser, access the Dashboard by using the external load balancer domain name or IP address defined by the `external_lb_vip_address` option in the `/etc/openstack_deploy/openstack_user_config.yml` file. The Dashboard uses HTTPS on port 443.
2. Authenticate by using the `admin` user name and the password defined by the `keystone_auth_admin_password` option in the `/etc/openstack_deploy/user_secrets.yml` file.

CHAPTER

SEVEN

NEXT STEPS

Now that you have verified your OpenStack cloud is operational, here is what you can do next:

7.1 Operate OpenStack-Ansible

Review our Operations Guide to learn about verifying your environment in more detail, and creating your first networks, images, and instances.

7.2 Contribute to OpenStack-Ansible

Review our Developer Documentation to learn about contributing to OpenStack-Ansible.