Oracle® OpenStack for Oracle Linux

Installation and User's Guide for Release 2

E64749-03
December 2015

Abstract

Document generated on: 2015-12-16 (revision: 264)

Table of Contents

Preface

Table of Contents

The preface contains information on how to use the Oracle OpenStack for Oracle Linux Installation and User's Guide, and an overview of what this guide contains.

OpenStack is a very flexible solution that can be configured, deployed, and used in various ways. This document provides guidelines to easily build a multi-node OpenStack deployment using Oracle Linux.

This document guides you through different options for deploying OpenStack on Oracle Linux. It begins by describing some basic features of OpenStack and describing how to set them up for an OpenStack deployment. It continues by providing the installation steps and demonstrating how to use several features on the network and storage stacks.

For simplicity, a small configuration has been chosen for examples. The example configuration includes a controller node, and multiple compute nodes. While it is possible to configure OpenStack in various ways with the tools described in this guide, the discussion is limited to a simplified configuration, allowing you to discover the possibilities of running Oracle Linux with OpenStack.

OpenStack has a rich feature set and various services. After going through this guide, you should have a good foundation from which to learn more about the different OpenStack features and experiment with them in the environment you create. OpenStack continues to evolve by adding new services and further developing existing services. To find out more about existing and new OpenStack features, see the OpenStack documentation available at:

http://docs.openstack.org/

1 Audience

The Oracle OpenStack for Oracle Linux Installation and User's Guide is intended for readers who would like to get started with OpenStack. No prior knowledge of OpenStack is required. Therefore, this guide can be used by those taking their first steps into the world of OpenStack. Using this guide, you can quickly put together an OpenStack deployment and test it to see whether Oracle Linux fits your requirements.

The guide is also useful for vendors who are interested in integrating with Oracle Linux, or who have customers interested in doing so. Vendors can use this guide to create a deployment on which integration with OpenStack and Oracle Linux can be tested.

2 Related Documents

For more information, see the following documents in the Oracle OpenStack for Oracle Linux documentation set:

- *Oracle OpenStack for Oracle Linux Release Notes*
- *Oracle OpenStack for Oracle Linux Installation and User's Guide*

You can also get the latest information on Oracle OpenStack for Oracle Linux at:

http://www.oracle.com/us/technologies/linux/openstack/

3 Command Syntax

Oracle Linux command syntax appears in `monospace` font. The dollar character (`$`), number sign (`#`), or percent character (`%`) are Oracle Linux command prompts. Do not enter them as part of the command. The following command syntax conventions are used in this guide:

Convention	**Description**

backslash `\`	A backslash is the Oracle Linux command continuation character. It is used in command examples that are too long to fit on a single line. Enter the command as displayed (with a backslash) or enter it on a single line without a backslash:
	`dd if=/dev/rdsk/c0t1d0s6 of=/dev/rst0 bs=10b \`
	`count=10000`

braces `{ }`	Braces indicate required items: `.DEFINE {macro1}`
brackets `[ ]`	Brackets indicate optional items: `cvtcrt` *`termname`* `[`*`outfile`*`]`
ellipses `...`	Ellipses indicate an arbitrary number of similar items: `CHKVAL fieldname value1 value2 ... valueN`
italics	Italic type indicates a variable. Substitute a value for the variable: *`library_name`*
vertical line `|`	A vertical line indicates a choice within braces or brackets: `FILE` *`filesize`* `[K|M]`

4 Conventions

The following text conventions are used in this document:

Convention	**Meaning**
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Chapter 1 Introduction to OpenStack

Table of Contents

This chapter gives an very high level introduction to the common components of OpenStack, terminology, architecture and network deployment options.

1.1 What Is OpenStack?

OpenStack is a cloud computing software platform that controls large pools of compute, storage, and networking resources in a data center. With OpenStack, you can manage different types of hypervisors, network devices and services, storage components, and more, using a single API that creates a unified data center fabric. OpenStack is, therefore, a pluggable framework that allows vendors to write plug-ins to implement a solution using their own technology, and which allows users to integrate their technology of choice.

1.2 OpenStack Services

To achieve this agility, OpenStack is built as a set of distributed services. These services communicate with each other, and are responsible for the various functions expected from virtualization/cloud management software. The following are some of the key OpenStack services:

- **Nova:** A compute service responsible for creating virtual machine instances and managing their life cycle, as well as managing the hypervisor of choice. The hypervisors are pluggable to Nova, while the Nova API remains the same, regardless of the underlying hypervisor.

- **Neutron:** A network service responsible for creating network connectivity and network services. It is capable of connecting with vendor network hardware via plug-ins. Neutron comes with a set of default services implemented by common tools. Network vendors can create plug-ins to replace any one of the services with their own implementation, adding value to their users.

- **Cinder:** A block storage service responsible for creating and managing external storage, including block devices and NFS. It is capable of connecting to vendor storage hardware through plug-ins. Cinder has several generic plug-ins, which can connect to NFS and iSCSI, for example. Vendors add value by creating dedicated plug-ins for their storage devices.

- **Swift:** An object and Binary Large Object (BLOB) storage service responsible for managing object-based storage.

- **Keystone:** An identity management system responsible for user and service authentication. Keystone is capable of integrating with third-party directory services such as LDAP.

- **Glance:** An image service responsible for managing images uploaded by users. Glance is not a storage service, but it is responsible for saving image attributes, making a virtual catalog of the images.
- **Heat:** An orchestration service responsible for managing the life cycle of the OpenStack infrastructure (such as servers, floating IP addresses, volumes, security groups, and so on) and applications. Uses Heat Orchestration Templates (HOT) to describe the infrastructure for an application and provides an API for Amazon's AWS template format.
- **Horizon:** A dashboard that creates a GUI for users to control the OpenStack deployment. This is an extensible framework to which vendors can add features. Horizon uses the same APIs exposed to users.
- **Murano:** An application catalog service for publishing cloud-ready applications from a catalog. An agent is installed into an instance's operating system, which enables deployment of the applications directly into the guest. Murano also includes a plug-in to the Horizon dashboard.

More details are available in the *OpenStack Cloud Administrator Guide* at:

http://docs.openstack.org/admin-guide-cloud/common/get_started_openstack_services.html

Important

Installation and deployment procedures provided in the core OpenStack documentation differ from those in this guide. Oracle OpenStack for Oracle Linux provides a set of packages which install a deployment toolkit. The toolkit performs the deployment of OpenStack services using Docker containers. For information on installing Oracle OpenStack for Oracle Linux, see Chapter 2, *Installing Oracle OpenStack for Oracle Linux*, and not the core OpenStack documentation.

OpenStack has many more services that are responsible for various features and capabilities, and the full list can be found on the OpenStack website at:

http://www.openstack.org/

The list provided here is limited to those needed to get started with Oracle OpenStack for Oracle Linux.

1.3 OpenStack Nodes

There are a number of node types used in OpenStack. Nodes are a physical host computer, with an operating system installed, with Oracle Linux using KVM (Kernel-based Virtual Machine) . The main node types we discuss in this guide are:

- A *controller node* is a system running Oracle Linux, and is where most of the OpenStack services are installed. The term controller node is used to discuss nodes that do not run virtual machine instances. The controller nodes may have all the non-compute services or only some of them. A controller node may also include the Oracle OpenStack for Oracle Linux toolkit, which is used to perform the deployment of OpenStack services to other nodes.
- A *compute node* is a system running Oracle Linux using KVM . A compute node runs the bare minimum of services to manage virtual machine instances.
- A *database node* is a system running Oracle Linux, and the services required to manage databases for images and instances.
- A *network node* is a system running Oracle Linux, and runs the neutron network worker daemon. The neutron worker daemon provides services such as providing an IP address to a booting Nova instance.

- A *storage node* is a system running Oracle Linux and the services required to manage storage for images and instances.
- A *master node* is a system running Oracle Linux, and the Oracle OpenStack for Oracle Linux toolkit used to deploy the OpenStack services to the nodes. A master node is not an OpenStack node, but the Oracle OpenStack for Oracle Linux toolkit may be installed on a controller node.

More detailed information on the node types is available in the *OpenStack Operations Guide* at:

http://docs.openstack.org/openstack-ops/content/example_architecture.html#node_types

1.4 OpenStack Instances

OpenStack virtual machines are called *instances*, mostly because they are instances of an image that is created upon request and that is configured when launched. The main difference between OpenStack and traditional virtualization technology is the way state is stored. With traditional virtualization technology, the state of the virtual machine is persistent.

OpenStack can support both *persistent* and *ephemeral* models. In the ephemeral model, an instance is launched from an image in the Image service, the image is copied to the run area, and when the copy is completed, the instance starts running. The size and connectivity of the instance are defined at the time of launching the instance. When an instance is terminated, the original image remains intact, but the state of the terminated instance is not retained. This ephemeral model is useful for scaling out quickly and maintaining agility for users.

In the persistent model, the instance is launched from a persistent volume on a compute node, or from a block storage volume, and not from the Image service. A volume can be any kind of persistent storage, including a file, a block device, an LVM partition, or any other form of persistent storage. In this case, when the instance is terminated, any session changes are retained and are present the next time an instance is launched. In the persistent model, the size and connectivity of the instance are also defined at the time the instance launches. In some sense, the persistent model in OpenStack is similar to the traditional approach to virtualization.

1.5 OpenStack Storage

The storage used in OpenStack can be either ephemeral or persistent. Ephemeral storage is deleted when an instance is terminated, while persistent storage remains intact. Persistent storage in OpenStack is referred to as a *volume*, regardless of the technology and device it is backed by. Persistent storage can either be used to launch an instance, or it can be connected to an instance as a secondary storage device to retain state. An example of this is a database launched as an ephemeral instance, with a volume connected to it to save the data. When the instance is terminated, the volume retains the data and can be connected to another instance as needed.

The OpenStack Cinder service is responsible for managing the volumes, and it offers a framework for vendors to create drivers. If a storage vendor wants to support OpenStack deployment and allow users to create volumes on the device, the vendor must create a Cinder driver that allows users to use the standard calls to control the storage device.

OpenStack also supports object storage using the Swift service.

1.6 OpenStack Networking

This section gives an introduction to networking in OpenStack.

1.6.1 Network Services

The OpenStack networking service, Neutron, offers a complete software-defined networking (SDN) solution, along with various network services. The network services Neutron can support include routing, firewall, DNS, DHCP, load balancing, VPN, and more.

Neutron, like Cinder, offers a framework for vendors to write plug-ins for various services. For example, a network vendor might want to offer a custom load balancer instead of the default load balancer provided by Neutron. The plug-in framework offers a powerful tool to build sophisticated network topologies using standard APIs.

1.6.2 Network Isolation: Tenant Networks

Tenant networks are the basis for Neutron's SDN capability. Neutron has full control of layer-2 isolation. This automatic management of layer-2 isolation is completely hidden from the user, providing a convenient abstraction layer required by SDN.

To perform the layer-2 separation, Neutron supports three layer-2 isolation mechanisms: VLANs, VxLANs, and GRE (Generic Routing Encapsulation) tunnels. You must define which mechanism should be used and set up the physical topology as required. Neutron is responsible for allocating the resources as needed. For example, you would configure the VLAN switch, allocate the VLAN range, and configure the VLAN in Neutron. When you define a new network, Neutron automatically allocates a VLAN and takes care of the isolation. You do not have to manage VLANs, and do not need to be aware of which VLAN was assigned to the network.

1.6.3 Complete Software-Defined Network Solution

OpenStack, using Neutron, presents a complete SDN solution. You can define isolated networks with any address space, and connect between those networks using virtual routers. You can define firewall rules without the need to touch or change any element of the physical network topology. Furthermore, there is a complete abstraction between the physical topology and the virtual networks, so that multiple virtual networks can share the same physical resources, without any security or address space concerns.

1.7 User Isolation: Multi-tenancy

Allowing multiple users to share the same physical environment while ensuring complete separation between them is a key feature of OpenStack. OpenStack is designed so that multiple tenants can share physical resources in a way that is transparent to the users. OpenStack offers ways to share virtual resources between tenants, but maintains complete separation where needed.

Chapter 2 Installing Oracle OpenStack for Oracle Linux

Table of Contents

This chapter gives an overview on the Oracle OpenStack for Oracle Linux deployment options. This chapter sets out how to set up the machines to perform the deployment (master nodes), and the steps to deploy Oracle OpenStack for Oracle Linux on the target nodes using Docker containers. This chapter also shows you how to set up and run the Oracle OpenStack for Oracle Linux deployment toolkit.

The required steps to install Oracle OpenStack for Oracle Linux are:

1. Set up the target nodes. The target nodes are the machines to be used for the various node types in an OpenStack deployment: controller, compute, database, network, and storage. Install and prepare Oracle Linux on each target node to be included in the deployment.

2. Set up the master node(s). The master nodes are used to perform the deployment of the Oracle OpenStack for Oracle Linux Docker containers on each target node. The Oracle OpenStack for Oracle Linux toolkit is installed on the master nodes to perform the deployment. The master nodes may be the same physical hosts used as controller nodes, or they may be separate machines.

3. Set up a Docker registry to host the Oracle OpenStack for Oracle Linux Docker images. These images are used by the master nodes to deploy Oracle OpenStack for Oracle Linux as Docker containers to each target node.

4. Perform the Oracle OpenStack for Oracle Linux deployment using the toolkit on a master node.

The examples in this guide use Oracle Linux Release 7 x86-64 hosts, using the repositories on the Oracle Linux Yum Server.

2.1 Oracle OpenStack for Oracle Linux Deployment Options

Oracle OpenStack for Oracle Linux supports various flexible deployment models, where each OpenStack service can be deployed separately on a different target node, or where services can be deployed together with other services. You perform this customized deployment using deployment groups. For more information on using deployment groups, see Section 4.1, “Using Deployment Groups”.

You can set up any number of nodes. Oracle OpenStack for Oracle Linux supports the following deployment models:

- **One controller node and one or more compute nodes:** This is a common deployment across multiple physical servers. In this case, all the control services are deployed on a node, while separate compute nodes are deployed for the sole purpose of running virtual machines. This deployment model is supported by Oracle.
- **One controller node, one network node, and one or more compute nodes:** Another common deployment configuration is when the network node is required to be separate from the rest of the services. This can be due to compliance or performance requirements. In this scenario, the network services are deployed to one node, and the rest of the management services are deployed on a separate controller node. Compute nodes can be deployed as required. This deployment model is supported by Oracle.
- **Multiple controller, network, storage, and compute nodes**. This is the model to enable High Availability (HA). To increase HA, you may consider adding more than two nodes for each node type. This deployment model is supported by Oracle.
- **All-in-one node:** A complete deployment of all the OpenStack services on one node. This deployment model is commonly used to get started with OpenStack or for development purposes. In this model, you have fewer configuration options, and the deployment does not require more than one node, so is well suited for deploying on a laptop, desktop, or even as a virtual machine. This deployment model is not supported by Oracle for production use.

2.2 Preparing Oracle Linux Target Nodes

A target node is any node that you intend to include in the Oracle OpenStack for Oracle Linux environment. Target nodes include controller, compute, database, network, and storage nodes. You must install Oracle Linux as the operating system for a target node. This section shows you how to set up the base operating system to support target nodes.

You can download the installation ISO of the latest version of Oracle Linux Release 7 from the Oracle Software Delivery Cloud at:

https://edelivery.oracle.com/linux

To prepare an Oracle Linux node:

1. Install Oracle Linux using the instructions in the *Oracle Linux Installation Guide for Release 7* at:

   http://docs.oracle.com/cd/E52668_01/E54695/html/

   Select **Minimal install** as the base environment.

   Important

   Create a BTRFS file system with at least 64GB available to host a local copy of the Docker registry images. On this file system, create a directory named `/var/lib/docker`.

2. If you enabled SELINUX during the install of the operating system, you must disable it in order for Docker to work with the BTRFS file system. Disable SELINUX by editing the `/etc/selinux/config` file to change the `SELINUX` option to *`disabled`*. Restart the operating system for the change to take effect:

   ```
   $ sudo reboot
   ```

3. Turn off the IP firewall, if it is running.

```
$ sudo systemctl stop firewalld
$ sudo systemctl disable firewalld
```

4. Ensure the node is configured to synchronize time using the Network Time Protocol (NTP).

   Time synchronization is essential to avoid errors with OpenStack operations.

   It is best to configure the controller nodes to synchronize the time from more accurate (lower stratum) NTP servers and to configure the other nodes to synchronize the time from the controller nodes. Ensure the time is synchronized on all nodes before deploying OpenStack.

   Information on network time configuration can be found in the *Oracle Linux Administration Guide for Release 7* at:
   http://docs.oracle.com/cd/E52668_01/E54669/html/ol7-nettime.html

5. Make sure your system is up-to-date:

   ```
   $ sudo yum upgrade
   ```

6. Enable the Oracle OpenStack for Oracle Linux repositories on the Oracle Linux Yum Server at http://public-yum.oracle.com. The Oracle OpenStack for Oracle Linux repository contains the Command Line Interface (CLI) and dependencies. To enable the repositories, edit the `/etc/yum.repos.d/public-yum-ol7.repo` file and set `enabled=1` in the following stanzas:

   - `ol7_optional_latest`
   - `ol7_addons`
   - `ol7_openstack20`

   Your configuration for these repositories should look like the following:

   ```
   ...
   [ol7_optional_latest]
   name=Oracle Linux $releasever Optional Latest ($basearch)
   baseurl=http://public-yum.oracle.com/repo/OracleLinux/OL7/optional/latest/$basearch/
   gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
   gpgcheck=1
   enabled=1

   [ol7_addons]
   name=Oracle Linux $releasever Add ons ($basearch)
   baseurl=http://public-yum.oracle.com/repo/OracleLinux/OL7/addons/$basearch/
   gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
   gpgcheck=1
   enabled=1
   ...
   [ol7_openstack20]
   name=OpenStack 2.0 packages for Oracle Linux 7 (x86_64)
   baseurl=http://public-yum.oracle.com/repo/OracleLinux/OL7/openstack20/x86_64/
   gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
   gpgcheck=1
   priority=20
   enabled=1
   ...
   ```

   If your `public-yum-ol7.repo` file does not include the `ol7_openstack20` stanza, you may not have the most up-to-date version of the repository file. To download the latest copy of the Oracle Linux Yum Server repository file:

   ```
   $ sudo curl -L -o /etc/yum.repos.d/public-yum-ol7.repo \
   ```

```
    http://public-yum.oracle.com/public-yum-ol7.repo
```

If you access the Yum repositories using a proxy server, update the `/etc/yum.conf` file with the proxy server address. For more information on using the Oracle Linux Yum Server, see:

http://docs.oracle.com/cd/E52668_01/E54669/html/ol7-yum.html

7. Install the Oracle OpenStack for Oracle Linux pre-installation package:

   ```
   $ sudo yum install openstack-kolla-preinstall
   ```

8. Set Docker to use BTRFS as the storage file system, and notify Docker that SELINUX is off. Edit the `/etc/sysconfig/docker` file to change:

   ```
   OPTIONS='--selinux-enabled'
   ```

   To

   ```
   OPTIONS='--storage-driver btrfs --selinux-enabled=false'
   ```

9. Restart Docker:

   ```
   $ sudo systemctl restart docker.service
   ```

The Oracle Linux node is ready to be included in an Oracle OpenStack for Oracle Linux deployment. See Section 2.5, “Deploying Oracle OpenStack for Oracle Linux” for information on using the Oracle OpenStack for Oracle Linux toolkit to perform the deployment.

2.3 Preparing Oracle Linux Master Nodes

A *master node* is a host from which you deploy Oracle OpenStack for Oracle Linux. You may want to use a controller node as a master node, or use a separate host. A master node is well suited to being installed on a controller node.

If you want to set up High Availability (HA) for the master node, you should set up the hosts with a shared file system mounted on `/etc/kolla`. You should also copy the SSH keys between the master nodes so they are identical.

This section shows you how to install the tools required on a master node to perform the deployment of Oracle OpenStack for Oracle Linux as Docker containers on the target nodes.

To prepare the master nodes:

1. Set up an Oracle Linux host as described in Section 2.2, “Preparing Oracle Linux Target Nodes”.

2. Install the Oracle OpenStack for Oracle Linux toolkit. The toolkit is installed using the Oracle OpenStack for Oracle Linux Yum repositories. If you do not have the repositories set up on the host, see Section 2.2, “Preparing Oracle Linux Target Nodes” for information on adding an enabling the repositories. To install the toolkit, enter:

   ```
   $ sudo yum install openstack-kollacli
   ```

   SSH keys for the deployment are created by the toolkit and the public key is copied to `/etc/kolla/kollacli/id_rsa.pub`.

3. A kolla group is created by the toolkit. The user you want to run the Oracle OpenStack for Oracle Linux toolkit CLI (kollacli) should be a member of the kolla group. To add an existing user to the kolla group, enter:

```
$ sudo usermod -aG kolla username
```

This user must log out and in again for the settings to take effect.

Important

To improve security, any CLI commands should be run as this user. Do not use root or the kolla user to run CLI commands.

The Oracle Linux host is ready to be used as a master node to perform the deployment of the Oracle OpenStack for Oracle Linux Docker containers on target nodes. See Section 2.5, "Deploying Oracle OpenStack for Oracle Linux" for information on using the Oracle OpenStack for Oracle Linux toolkit to perform the deployment.

2.4 Setting up a Docker Registry

A Docker registry is required to host the Oracle OpenStack for Oracle Linux Docker images used during the deployment. The Docker images are included in the Oracle OpenStack for Oracle Linux zip file, which you can download from the Oracle Software Delivery Cloud at:

https://edelivery.oracle.com/linux

For detailed information on installing and using the Docker Engine on Oracle Linux see the *Oracle Linux Administrator's Guide for Release 7* at:

http://docs.oracle.com/cd/E52668_01/E54669/html/ol7-docker.html

You may want to use a master node to host the Docker registry, or a separate host. This guide sets out the steps to set up a registry to give you maximum flexibility and choice as to where to host the registry. The machine that hosts the Docker registry must have a connection to the public network to download a copy of the main Docker registry on the Internet.

Caution

By default, the Docker registry uses port 5000 and the following instructions use this port. However, port 5000 is also used by the Keystone service which runs on controller nodes. If you set up the Docker registry on a controller node, use a different free port for the Docker registry, for example port 5443.

This section shows you how to set up a Docker registry to host the Docker containers for the Oracle OpenStack for Oracle Linux deployment.

To set up a Docker registry:

1. The Docker registry requires 15GB of storage in `/var/lib/registry`. You can either create the directory on your main partition, or create a separate file system and mount it as `/var/lib/registry`. For improved performance, use BTRFS.

2. If you are setting up a Docker registry on a machine that does not already have Docker installed, install the Docker engine package. You do not need to perform this step if you have installed the Oracle OpenStack for Oracle Linux toolkit on the host, as described in Section 2.3, "Preparing Oracle Linux Master Nodes".

   ```
   $ sudo yum install docker-engine
   ```

3. Start the Docker service:

   ```
   $ sudo systemctl start docker.service
   ```

Optionally, if you want the service to start automatically on boot, enter:

```
$ sudo systemctl enable docker.service
```

4. If your host is behind a firewall, you should set up a proxy for the Docker service. Information on setting up a proxy for Docker is in *Oracle Linux Administrator's Guide*:

   https://docs.oracle.com/cd/E52668_01/E54669/html/section_kfy_f2z_fp.html

5. To enable authentication with the Docker registry, add your CA certificate to Docker. Replace *`registry_hostname`* with the fully qualified domain name (FQDN) of the machine hosting the Docker registry. The registry *`port`* is assumed to be running on port 5000, but yours may be running on port 5443, or some other number. The final command here assumes *`path`* is the location on the host of the CA certificate for your domain. The final command copies and renames the domain certificate to the location required by Docker.

   ```
   $ mkdir -p /etc/docker/certs.d/registry_hostname:port
   $ cp path/domain.crt /etc/docker/certs.d/registry_hostname:port/ca.crt
   ```

   If you want to use a self-signed CA certificate, you must distribute the public certificate to **all** master and target nodes, in the location:

   `/etc/docker/certs.d/registry_hostname:port/ca.crt`

   For example, to create a self-signed certificate:

   ```
   $ mkdir -p /var/lib/registry/conf.d
   $ cd /var/lib/registry/conf.d
   $ openssl req -newkey rsa:4096 -nodes -sha256 -keyout domain.key -x509 -days 365 -out domain.crt
   $ chmod 600 domain.key
   $ mkdir -p /etc/docker/certs.d/registry_hostname:port
   $ cp /var/lib/registry/conf.d/domain.crt /etc/docker/certs.d/registry_hostname:port/ca.crt
   ```

   Make sure the domain name you use when creating the CA certificate is identical to the FQDN of the Docker registry host.

   Copy the `ca.crt` file to all master and target nodes to the `/etc/docker/certs.d/registry_hostname:port/` directory. This step is *not* required when using a verified CA certificate, it is only required for self-signed certificates.

   ```
   $ sudo scp /etc/docker/certs.d/registry_hostname:port/ca.crt \
     root@myremotenode:/etc/docker/certs.d/registry_hostname:port/ca.crt
   ```

6. Create a Docker registry. Replace port with the *`port`* number for your registry (default is 5000).

   ```
   $ sudo docker run -d -p port:5000 --name registry --restart=always \
       -v /var/lib/registry:/registry_data \
       -e REGISTRY_STORAGE_FILESYSTEM_ROOTDIRECTORY=/registry_data \
       -e REGISTRY_HTTP_TLS_KEY=/registry_data/conf.d/domain.key \
       -e REGISTRY_HTTP_TLS_CERTIFICATE=/registry_data/conf.d/domain.crt \
       registry:2
   ```

   The Docker registry is created, and the base Docker registry images are pulled into the new registry.

7. Restart Docker:

   ```
   $ sudo systemctl restart docker.service
   ```

   If you are using self-signed CA certificates, you should also reload and restart Docker on each target node.

8. On the host where you downloaded the Oracle OpenStack for Oracle Linux Docker images zip file (from the Oracle Software Delivery Cloud), install the decompression packages, then decompress the file:

   ```
   $ sudo yum install unzip bzip2
   $ unzip ol-openstack-images-version.zip
   ```

   The following files are unzipped to the directory:

   - `import_to_registry.sh`: A script to load the images to the Docker registry.
   - `ol-openstack-images-version.sha256sum`: A sha256sum hash file.
   - `ol-openstack-images-version.tar.bz2`: Contains the Docker registry files. You do not need to decompress this file.

9. Load the Oracle OpenStack for Oracle Linux Docker images from the `ol-openstack-images-version.tar.bz2` file to the Docker registry using the loader script:

   ```
   $ sudo ./import_to_registry.sh registry_hostname:port
   ```

   If you want to clean up the local registry storage on the host performing the registry load, use the `-c` option, for example:

   ```
   $ sudo ./import_to_registry.sh -c registry_hostname:port
   ```

   The Docker registry loading may take some time.

2.5 Deploying Oracle OpenStack for Oracle Linux

To perform an Oracle OpenStack for Oracle Linux deployment, you must first set up your Oracle Linux target nodes. See Section 2.2, “Preparing Oracle Linux Target Nodes” for information on setting up the target nodes.

You must also set one or more *master* nodes from which to run the CLI to perform the deployment. A master node includes the Oracle OpenStack for Oracle Linux toolkit which performs the deployment of the Docker containers to the target nodes. A master node may be set up on a controller node, or it may be a separate host.

The master node logs into target node (controller, compute, storage, and/or network node), and performs the deployment and configuration of the OpenStack services using Docker containers. For information on setting up the master nodes, see Section 2.3, “Preparing Oracle Linux Master Nodes”.

You must also set up a Docker registry to host the Oracle OpenStack for Oracle Linux Docker images. See Section 2.4, “Setting up a Docker Registry” for information on setting up a Docker registry.

This section shows you how to use the Oracle OpenStack for Oracle Linux toolkit on a master node to perform the Oracle OpenStack for Oracle Linux deployment to the target nodes.

For detailed information on the toolkit CLI and its commands, see Appendix A, *Oracle OpenStack for Oracle Linux Toolkit CLI Reference*.

To perform the deployment from a master node:

1. Set up the passwords needed for the deployment using the `kollacli password set` command.

Important

If you do not set the passwords, the defaults are used. In a production environment, you should set password values for all password names.

You are prompted for each password value. The password value is not displayed on screen.

```
$ kollacli password set cinder_database_password
$ kollacli password set cinder_keystone_password
$ kollacli password set database_password
$ kollacli password set docker_registry_password
$ kollacli password set glance_database_password
$ kollacli password set glance_keystone_password
$ kollacli password set heat_database_password
$ kollacli password set heat_keystone_password
$ kollacli password set heat_domain_admin_password
$ kollacli password set keystone_admin_password
$ kollacli password set keystone_database_password
$ kollacli password set murano_database_password
$ kollacli password set murano_keystone_password
$ kollacli password set nova_database_password
$ kollacli password set nova_keystone_password
$ kollacli password set neutron_database_password
$ kollacli password set neutron_keystone_password
$ kollacli password set rabbitmq_password
$ kollacli password set swift_keystone_password
```

2. Add each host (target node) to the deployment using the `kollacli host add` command, for example:

   ```
   $ kollacli host add hostname
   ```

3. Copy the SSH public key for the kolla user to each host and confirm that the kolla user can log in to the host using SSH keys.

   Use the `kollacli host setup` command to perform this step. This step needs only be performed once, and should not be required again unless the SSH keys for the kolla user are changed. For example:

   ```
   $ kollacli host setup hostname
   ```

   You are prompted for the password value. By default, the password is for the root user on the target node. This may not always be desirable. If you want to change the default user, set the `KOLLA_CLI_SETUP_USER` environment variable before running the `kollacli host setup` command. The variable should contain the user name of an alternative user on the target host. The user must already exist on the target host and have sufficient privileges to be able to write to the `/usr/share/kolla/.ssh/authorized_keys` file, which is owned by the kolla user and group.

   If you want to set up multiple hosts with a single command, you can create a YAML file that contains the names of the hosts to set up and the passwords to use. If the file specifies a user name for a host (optional), this is used instead of root or the user name in the `KOLLA_CLI_SETUP_USER` environment variable. For example, you might have a file named `hosts.yml` in the `/myhome` directory which contains:

   ```
   ---
   mycontrolnode1.example.com:
       password: password
   mycontrolnode2.example.com:
       password: password
   mycomputenode1.example.com:
       password: password
   ```

```
    uname: user_name
mycomputenode2.example.com:
    password: password
    uname: user_name
mycomputenode3.example.com:
    password: password
    uname: user_name
mycomputenode4.example.com:
    password: password
    uname: user_name
```

You could then set up all hosts with the command:

```
$ kollacli host setup --file /myhome/hosts.yml
```

Caution

For security reasons, make sure you secure the file containing these login credentials.

4. Assign each host to a deployment group. Deployment groups are used to allocate one or more hosts to a group, so they have the same services installed. A deployment group directly maps to one of the deployment scenarios listed in Section 2.1, “Oracle OpenStack for Oracle Linux Deployment Options”. The default groups are control, compute, database, network, and storage. You can include a host in more than one group. Use the `kollacli group addhost` command to perform this step. In this example, there are two controller nodes, two database nodes, two network nodes, two storage nodes, and four compute nodes.

   ```
   $ kollacli group addhost control mycontrolnode1.example.com
   $ kollacli group addhost control mycontrolnode2.example.com
   $ kollacli group addhost database mycontrolnode1.example.com
   $ kollacli group addhost database mycontrolnode2.example.com
   $ kollacli group addhost network mycontrolnode1.example.com
   $ kollacli group addhost network mycontrolnode2.example.com
   $ kollacli group addhost storage mycontrolnode1.example.com
   $ kollacli group addhost storage mycontrolnode2.example.com
   $ kollacli group addhost compute mycomputenode1.example.com
   $ kollacli group addhost compute mycomputenode2.example.com
   $ kollacli group addhost compute mycomputenode3.example.com
   $ kollacli group addhost compute mycomputenode4.example.com
   ```

   For more information on using deployment groups, see Section 4.1, “Using Deployment Groups”.

5. Set the Docker registry properties for your registry. The host name or IP address is the address of the server that hosts your Docker registry. Use the `kollacli property set` command to perform this step.

   ```
   $ kollacli property set docker_registry registry_hostname:port
   ```

6. Configure the network properties.

   Use the `kollacli property set` command to set the following properties:

   - `network_interface`: The name of the network interface (for example, `em1`) on **all** nodes connected to the internal management network (where OpenStack services listen for connections). All nodes in the deployment must have an IPv4 address on this interface. The internal management network should be a private network. By default, this interface is also used for the VXLAN tunnel and storage network traffic.

     If the nodes have differing network interface names, create a bond on all hosts and assign an interface to the bond, and then use the bond name for the `network_interface`.

- `kolla_internal_address`: The IP address on the internal management network used to access OpenStack control services.

  For a non-HA deployment, this is the IP address of the `network_interface` on the controller node.

  For a HA deployment, this is an unused IP address on the internal management network and is used by Keepalived as the virtual IP (VIP) address. Keepalived maps the VIP address to the real IP address of the controller node that runs the current active HAProxy.

  This value is used to populate part of each OpenStack service's `private_url` and `admin_url` in Keystone.

- `kolla_external_address`: The DNS name or IP address to be used in the public OpenStack API endpoints. By default, this property is set to the same value as `kolla_internal_address`.

  For private cloud deployments, do not change the default setting (the `kolla_internal_address` is used).

  For public cloud deployments, this is a DNS name that resolves to a public IP address. The OpenStack operator is responsible for mapping the public IP address to the `kolla_internal_address`, for example through a firewall.

  This value is used to populate part of each OpenStack service's `public_url` in Keystone.

- `neutron_external_interface`: The name of the network interface (for example, em2) on all **network** nodes which is connected to the external network where the neutron public network will be created. This interface should not have an IP address and should not be the same as the `network_interface`.

  If the network nodes have differing network interface names, create a bond on the network nodes and assign an interface to the bond, and then use the bond name for the `neutron_external_interface`.

For example, in a HA deployment where all nodes are connected on the network interface named em1, and a virtual IP address of `10.0.0.20` is used for HA on this network, and all the network nodes are connected to an external network on an interface named em2:

```
$ kollacli property set network_interface em1
$ kollacli property set kolla_internal_address 10.0.0.20
$ kollacli property set neutron_external_interface em2
```

7. (Optional) The default tenant networking option is to use GRE/VxLANs. GRE/VxLANs are the preferred tenant networking option for enterprise deployments. If you want to use GRE/VxLAN networks for the tenant networks, you do not need to complete this step. If you want to use VLANs for tenant networks, you can set this up as part of the deployment configuration. The properties you need to set are:

   - `neutron_tenant_type`: The tenant network type. Valid options for this are `vlan`, `gre` and `vxlan`. The default is `vxlan`. To use VLANs for tenant networks, set this to `vlan`.

   - `neutron_vlan_physnet`: The name of the VLAN network. The default is `physnet1`, which is generally used to name flat networks. To avoid confusion with a flat network, you should change this to something other than the default, for example, `physnet2`.

   - `neutron_vlan_range`: The range for VLAN IDs, in the format *`start_range`*`:`*`end_range`*. The default range is `1:1000`.

- `neutron_vlan_bridge`: The name for the VLAN network bridge. The default name is `br-vlan`.
- `neutron_vlan_interface`: The VLAN traffic network interface name. The network interface must be available on each compute and network node, and must have the same name. The interface must not have an IP address (because it is a bridged interface) and it must not be the same interface as either the `network_interface` or `neutron_external_interface`.

For example:

```
$ kollacli property set neutron_tenant_type vlan
$ kollacli property set neutron_vlan_physnet physnet2
$ kollacli property set neutron_vlan_range 1000:2000
$ kollacli property set neutron_vlan_interface em3
```

8. If you are performing an all-in-one deployment for testing purposes, disable HAProxy and set the deployment to be local (instead of the default of remote):

```
$ kollacli property set enable_haproxy no
$ kollacli setdeploy local
```

9. Deploy the Docker containers for Oracle OpenStack for Oracle Linux to the target nodes:

```
$ kollacli deploy
```

This command deploys all the containers to all the hosts. You can fine tune the deployment with further options as required. You can deploy specific containers to some or all hosts, or to deployment groups. For more information on the deployment options, see the `kollacli deploy` command.

You can view the Docker deployment logs using the `docker logs` command, which is part of the Docker CLI. For information on using the `docker logs` command, see the Docker documentation at:

https://docs.docker.com/reference/commandline/logs/

Oracle OpenStack for Oracle Linux is deployed to each configured node and ready for use.

Example 2.1 Multi-node HA Deployment

This example shows a multi-node deployment and it is assumed all password values have been set. This deployment uses two hosts to perform the roles of controller, database, network and storage, and four hosts as compute nodes. This is deployment enables HA as there are failover nodes for each node type. All nodes are connected on the em1 network interface, a virtual IP address of `10.0.0.20` is used for HA on this network, and all the network nodes are connected to an external network on the em2 network interface:

```
$ kollacli setdeploy remote
$ kollacli host add mycontrolnode1.example.com
$ kollacli host add mycontrolnode2.example.com
$ kollacli host add mycomputenode1.example.com
$ kollacli host add mycomputenode2.example.com
$ kollacli host add mycomputenode3.example.com
$ kollacli host add mycomputenode4.example.com
$ kollacli host setup mycontrolnode1.example.com
$ kollacli host setup mycontrolnode2.example.com
$ kollacli host setup mycomputenode1.example.com
$ kollacli host setup mycomputenode2.example.com
$ kollacli host setup mycomputenode3.example.com
$ kollacli host setup mycomputenode4.example.com
$ kollacli group addhost control mycontrolnode1.example.com
$ kollacli group addhost control mycontrolnode2.example.com
$ kollacli group addhost compute mycomputenode1.example.com
$ kollacli group addhost compute mycomputenode2.example.com
$ kollacli group addhost compute mycomputenode3.example.com
$ kollacli group addhost compute mycomputenode4.example.com
$ kollacli group addhost database mycontrolnode1.example.com
$ kollacli group addhost database mycontrolnode2.example.com
$ kollacli group addhost network mycontrolnode1.example.com
$ kollacli group addhost network mycontrolnode2.example.com
$ kollacli group addhost storage mycontrolnode1.example.com
$ kollacli group addhost storage mycontrolnode2.example.com
$ kollacli property set docker_registry myregistry.example.com:5000
$ kollacli property set network_interface em1
$ kollacli property set kolla_internal_address 10.0.0.20
$ kollacli property set neutron_external_interface em2
$ kollacli deploy
```

Example 2.2 Single Node (All-in-One) Deployment

This example shows single node (all-in-one) deployment and it is assumed all password values have been set. One host is used for all roles. The node's IP address is `10.0.0.10` on the em1 network interface, and has a connection to an external network on the em2 network interface. The docker registry is also configured on this node and uses port 5443.

HA is not available in this deployment type. This may be useful for testing in a virtual machine.

The all-in-one deployment option can only be used for testing. It should not be used in a production environment, and is not supported by Oracle.

```
$ kollacli setdeploy local
$ kollacli host add mynode.example.com
$ kollacli group addhost control mynode.example.com
$ kollacli group addhost compute mynode.example.com
$ kollacli group addhost database mynode.example.com
$ kollacli group addhost network mynode.example.com
$ kollacli group addhost storage mynode.example.com
$ kollacli property set docker_registry mynode.example.com:5443
$ kollacli property set enable_haproxy no
$ kollacli property set network_interface em1
$ kollacli property set kolla_internal_address 10.0.0.10
$ kollacli property set neutron_external_interface em2
$ kollacli deploy
```

2.6 Updating Oracle OpenStack for Oracle Linux

Updating an Oracle OpenStack for Oracle Linux environment includes updating the Oracle OpenStack for Oracle Linux packages, and updating and deploying the updated Docker containers. It does not include updates to container schemas or container API changes. This section discusses the procedures to update the OpenStack packages and containers.

2.6.1 Updating Oracle OpenStack for Oracle Linux Packages

The Oracle OpenStack for Oracle Linux packages can be updated using the Oracle Linux Yum Server or the Oracle Unbreakable Linux Network (ULN). To update the Oracle OpenStack for Oracle Linux packages, enter:

```
$ sudo yum upgrade openstack-kolla*
```

This updates the following packages, if they are installed:

- openstack-kolla
- openstack-kolla-ansible
- openstack-kolla-preinstall

- openstack-kolla-utils
- openstack-kollacli

2.6.2 Updating Oracle OpenStack for Oracle Linux Docker Containers

An update may involve new versions of the Oracle OpenStack for Oracle Linux Docker containers.

You update the containers using the `kollacli deploy` command. For information and examples of upgrading containers, see Section A.1.1.2, “deploy”. This section outlines a high level procedure for updating containers in an HA and non-HA environment.

To update OpenStack containers:

1. Download the updated OpenStack Docker containers from the Oracle Software Delivery Cloud. For the download location of the OpenStack Docker containers, see Getting the Software in the *Oracle OpenStack for Oracle Linux Release Notes*.
2. Unzip the downloaded file, and import the updated OpenStack containers to your Docker registry. For information on loading the Docker containers to your registry, see Section 2.4, “Setting up a Docker Registry”.
3. Shut down all running instances in the deployment.
4. Use the `kollacli deploy --serial` command to update the containers.

   Currently, the `--serial` option is the only supported method for updating. The `--services`, `--groups`, and `--hosts` options must only be used with new deployments.

   By default, the data containers for the various OpenStack services are preserved when you update.

   For more information, see `kollacli deploy` command.
5. Reboot all network and compute nodes. For networking to function after the update, you must reboot these node types.

2.7 Setting up Storage

This section discusses setting up the storage for an Oracle OpenStack for Oracle Linux deployment.

2.7.1 Adding Block Storage Drivers for Cinder

The LVMVolumeDriver is the default Cinder driver. You can add additional block storage drivers to Cinder as needed. To do this, you need to enable the driver in the Cinder configuration file. On the master node, add an entry to the `/etc/kolla/config/cinder.conf` file using the format:

```
volume_driver = cinder.volume.drivers.driver_name
```

If the file does not exist, create it. For example, to enable the Oracle OpenStack Cinder Driver for Oracle Flash Storage Systems, use:

```
volume_driver = cinder.volume.drivers.ofs.ofs.OracleFSFibreChannelDriver
```

To download the Oracle OpenStack Cinder Driver for Oracle Flash Storage Systems, go to:

http://www.oracle.com/technetwork/server-storage/san-storage/downloads/index.html

For more information about OpenStack block storage drivers, see the *OpenStack Configuration Reference*:

http://docs.openstack.org/kilo/config-reference/content/section_volume-drivers.html

2.7.2 Using Glance in an HA Deployment

Glance uses the local file system to store virtual machine images. Glance does not support HA out of the box. If you want to use HA with Glance, you can:

- Use an NFS file system mounted on `/var/lib/glance/images` on each Glance node.
- Use Swift as the storage for Glance.
- Deploy Glance to only one host.

To assist you in deploying Glance on more than one node (in an HA environment) a directory is created in the ol-openstack-glance-api container for each host on which Glance is deployed. This directory can be used as the mount point for your NFS share. The directory is located in `/var/lib/glance/images`. You must manually mount this directory to an NFS share or shared file system on each Glance node.

You may want instead to use Swift as the storage method for Glance. This fully enables HA for Glance.

Alternatively, you can deploy Glance to a single node. This avoids any HA issues with file-based storage, though it also means that Glance is not HA-enabled. To deploy Glance to a single node, create a deployment group which contains only Glance, and add one host to that group.

2.7.3 Setting up Swift

Swift is disabled by default. In order to use object-based storage, you need to first create Swift rings, and enable Swift. The steps in this procedure are to be performed on a master node.

To enable Swift:

1. Create and rebalance the Swift ring manually.

   You can use the following example configuration scripts as the basis for your Swift ring configuration.

   The scripts create the following required files in the `/etc/kolla/config/swift` directory on the master node:

   - `account.builder`
   - `account.ring.gz`
   - `container.builder`
   - `container.ring.gz`
   - `object.builder`
   - `object.ring.gz`

   Example 2.3 Swift multi-node ring configuration script

   The following script is an example of setting up the Swift ring for a multi-node deployment. The example uses a three-node storage array.

   The `REGISTRY` and `TAG` variables use the Kolla properties that contain the location and port of your OpenStack Docker registry and the OpenStack release number.

   Set the IP address for each storage node using the *`node_IP`* option in the `STORAGE` array at the start of the script, for example, `STORAGE[1]=192.168.0.10`.

```
#!/usr/bin/env bash

REGISTRY=registry_hostname:registry_port
TAG=openstack_release

STORAGE[1]=node_IP
STORAGE[2]=node_IP
STORAGE[3]=node_IP

# Note: In this example, each storage node is placed in its own zone

# Object ring
docker run --rm \
  -v /etc/kolla/config/swift/:/etc/kolla/config/swift/ \
  ${REGISTRY}/oracle/ol-openstack-swift-base:${TAG} \
  swift-ring-builder /etc/kolla/config/swift/object.builder create 10 3 1

for NODE in 1 2 3; do
  echo "object.builder: Adding ${STORAGE[$NODE]} to zone z${NODE}"
  docker run --rm \
    -v /etc/kolla/config/swift/:/etc/kolla/config/swift/ \
    ${REGISTRY}/oracle/ol-openstack-swift-base:${TAG} swift-ring-builder \
    /etc/kolla/config/swift/object.builder add z${NODE}-${STORAGE[$NODE]}:6000/sdb1 1
done

# Account ring
docker run --rm \
  -v /etc/kolla/config/swift/:/etc/kolla/config/swift/ \
  ${REGISTRY}/oracle/ol-openstack-swift-base:${TAG} \
  swift-ring-builder /etc/kolla/config/swift/account.builder create 10 3 1

for NODE in 1 2 3; do
  echo "account.builder: Adding ${STORAGE[$NODE]} to zone z${NODE}"
  docker run --rm \
    -v /etc/kolla/config/swift/:/etc/kolla/config/swift/ \
    ${REGISTRY}/oracle/ol-openstack-swift-base:${TAG} swift-ring-builder \
    /etc/kolla/config/swift/account.builder add z${NODE}-${STORAGE[$NODE]}:6001/sdb1 1
done

# Container ring
docker run --rm \
  -v /etc/kolla/config/swift/:/etc/kolla/config/swift/ \
  ${REGISTRY}/oracle/ol-openstack-swift-base:${TAG} \
  swift-ring-builder /etc/kolla/config/swift/container.builder create 10 3 1

for NODE in 1 2 3; do
  echo "container.builder: Adding ${STORAGE[$NODE]} to zone z${NODE}"
  docker run --rm \
    -v /etc/kolla/config/swift/:/etc/kolla/config/swift/ \
    ${REGISTRY}/oracle/ol-openstack-swift-base:${TAG} swift-ring-builder \
    /etc/kolla/config/swift/container.builder add z${NODE}-${STORAGE[$NODE]}:6002/sdb1 1
done

for ring in object account container; do
  docker run --rm \
    -v /etc/kolla/config/swift/:/etc/kolla/config/swift/ \
    ${REGISTRY}/oracle/ol-openstack-swift-base:${TAG} swift-ring-builder \
    /etc/kolla/config/swift/${ring}.builder rebalance
done
```

Example 2.4 Swift all-in-one ring configuration script

The following script is an example of setting up the Swift ring for an all-in-one deployment. Only use this script for testing, as an all-in-one deployment is not supported for a production system.

The `REGISTRY` and `TAG` variables use the Kolla properties that contain the location and port of your OpenStack Docker registry and the OpenStack release number.

The `NODE_IP` variable uses the Kolla property that contains the host's IP address on the internal management network.

```
#!/usr/bin/env bash

REGISTRY=registry_hostname:registry_port
TAG=openstack_release
NODE_IP=kolla_internal_address

# Object ring
docker run --rm \
  -v /etc/kolla/config/swift/:/etc/kolla/config/swift/ \
  ${REGISTRY}/oracle/ol-openstack-swift-base:${TAG} \
  swift-ring-builder /etc/kolla/config/swift/object.builder create 10 3 1

for partition in sdb1 sdb2 sdb3; do
  docker run --rm \
    -v /etc/kolla/config/swift/:/etc/kolla/config/swift/ \
    ${REGISTRY}/oracle/ol-openstack-swift-base:${TAG} swift-ring-builder \
    /etc/kolla/config/swift/object.builder add z1-${NODE_IP}:6000/${partition} 1
done

# Account ring
docker run --rm \
  -v /etc/kolla/config/swift/:/etc/kolla/config/swift/ \
  ${REGISTRY}/oracle/ol-openstack-swift-base:${TAG} \
  swift-ring-builder /etc/kolla/config/swift/account.builder create 10 3 1

for partition in sdb1 sdb2 sdb3; do
  docker run --rm \
    -v /etc/kolla/config/swift/:/etc/kolla/config/swift/ \
    ${REGISTRY}/oracle/ol-openstack-swift-base:${TAG} swift-ring-builder \
    /etc/kolla/config/swift/account.builder add z1-${NODE_IP}:6001/${partition} 1
done

# Container ring
docker run --rm \
  -v /etc/kolla/config/swift/:/etc/kolla/config/swift/ \
  ${REGISTRY}/oracle/ol-openstack-swift-base:${TAG} \
  swift-ring-builder /etc/kolla/config/swift/container.builder create 10 3 1

for partition in sdb1 sdb2 sdb3; do
  docker run --rm \
    -v /etc/kolla/config/swift/:/etc/kolla/config/swift/ \
    ${REGISTRY}/oracle/ol-openstack-swift-base:${TAG} swift-ring-builder \
    /etc/kolla/config/swift/container.builder add z1-${NODE_IP}:6002/${partition} 1
done

for ring in object account container; do
  docker run --rm \
    -v /etc/kolla/config/swift/:/etc/kolla/config/swift/ \
    ${REGISTRY}/oracle/ol-openstack-swift-base:${TAG} swift-ring-builder \
    /etc/kolla/config/swift/${ring}.builder rebalance
done
```

2. Enable Swift using:

```
$ kollacli property set enable_swift yes
```

For more information on configuring Swift, see the OpenStack Kolla documentation at:

http://docs.openstack.org/developer/kolla/swift-readme.html

2.7.4 Setting the ISCSI Initiator Name

You can set the name of the ISCSI initiator, rather than use the default. If you do not set the name, one is generated for you during the deployment. This is an optional setting. You can set the ISCSI initiator name in the `/etc/kolla/nova-iscsid/initiatorname.iscsi` file on each compute node. If the `initiatorname.iscsi` file does not exist, create it. The file should include one line, with the name of the initiator, in the format:

```
InitiatorName=iqn.yyyy-mm.naming-authority:unique_name
```

An example file might be:

```
InitiatorName=iqn.1988-12.com.oracle:myiscsihost
```

Chapter 3 Using OpenStack CLIs

Table of Contents

In order to access the OpenStack command line tools, such as `nova`, `glance`, and so on, you need to install the python-*`openstack_project`* client packages. Installing these packages may cause dependency problems with Docker and OpenStack client package versions. To make the installation of the OpenStack client tools easier, Oracle provides a single package for you which includes all the OpenStack CLI tools. The package name is openstack-kolla-utils . This package is provided with pre-tested package dependencies, and enables you to submit OpenStack CLI commands via Docker to an OpenStack service in a container. This toolkit is a single package you can install, with all the OpenStack Docker command line tool wrappers.

Oracle suggests you install the openstack-kolla-utils package on a master node, though you can install it on any Oracle Linux Release 7 host.

3.1 Installing the OpenStack CLI Docker Toolkit

This section shows you how to install the OpenStack CLI Docker Toolkit.

To install the Docker Toolkit:

1. Set up an Oracle Linux host as described in Section 2.3, “Preparing Oracle Linux Master Nodes”.

2. Install the OpenStack Docker CLI Toolkit:

   ```
   $ sudo yum install openstack-kolla-utils
   ```

3. Optionally, set the environment variables to use with the Toolkit:

   ```
   $ export variable_name=variable_value
   ```

   The variables used by the Docker Toolkit are listed in the following table:

Environment Variable	Purpose
`OS_AUTH_URL`	Authentication URL.
`OS_TENANT_NAME`	Keystone tenant name.
`OS_USERNAME`	Keystone user name.
`OS_PASSWORD`	Keystone password.
`OS_PROJECT_NAME`	Keystone project name.
`OS_PROJECT_DOMAIN_ID`	Keystone domain ID containing the project.
`OS_USER_DOMAIN_ID`	Keystone user's domain ID.
`ENV_FILE`	The location and name of a file which contains key/value pairs. Used to pass additional environment variables to the toolkit. Each key/value pair should be on a new line.
`OPENSTACK_UTILS_IMAGE`	Docker registry image name. The default is `oracle/ol-openstack-utils:latest`. You should not need to change this.

The list of variables in this table is not a complete list. For more information on OpenStack Command Line variables, and a full list of variables you can use, see the *OpenStack Keystone Command Line Utility* documentation at:

http://docs.openstack.org/developer/python-keystoneclient/man/keystone.html

3.2 Using the OpenStack CLI Docker Toolkit

To use the OpenStack CLI Docker Toolkit, use the syntax:

Syntax

`docker-ostk` [*`openstack_client`* | *`openstack_client_argument`*] [`-h` | `--help`]

Description

The OpenStack Docker toolkit wraps `docker run` commands, and includes the environment variables you set. So instead of needing to issue a command like:

```
$ docker run --rm -it --env ADMIN_USER=kolla --env ADMIN_PASS=password openstack-utils nova list
```

You can use the more straight-forward command of:

```
$ docker-ostk nova list
```

For a list of OpenStack CLI commands and their arguments, see the *OpenStack Command-Line Interface Reference*:

http://docs.openstack.org/cli-reference/content/

The first time you issue a `docker-ostk` command, the toolkit checks the Docker registry for a local copy of the ol-openstack-utils image. If a local copy is not available, a copy is downloaded to your local copy of the registry.

Note

The Docker registry location should be set using the `kollacli property set docker_registry` command. Do not manually edit the `/etc/kolla/globals.yml` file to set the Docker registry location.

To allow access to local files (for example, Glance images or Nova key pairs), the current working directory is bind-mounted into the container as `/data`. When you issue commands that require access to the container's file system, you must prefix the local file name with `/data`, for example:

```
$ docker-ostk nova keypair-add --pub-key /data/demokey.pub demokey
```

Options

The following table shows the available options for this command.

Option	Description
`openstack_client`	An OpenStack CLI client, for example, `nova`.
`openstack_client_argument`	An OpenStack CLI client command and its arguments, for example, `list`.

Option	Description
[-h | --help]	Displays help on syntax and usage for this command.

Examples

Example 3.1 To list the Nova instances:

```
$ docker-ostk nova list
```

Example 3.2 To start a Nova instance:

```
$ docker-ostk nova boot --boot-volume 2875856e-7595-4acb-b70c-0f6b3d4b19bf --flavor 1 ol7 \
  --nic net-id=c61c4043-5b2b-46ce-8316-e35e7daee378
```

Example 3.3 To list the Glance images:

```
$ docker-ostk glance image-list
```

Chapter 4 Using the Oracle OpenStack for Oracle Linux Toolkit

Table of Contents

The Oracle OpenStack for Oracle Linux toolkit is a Command Line Interface (CLI) based on the Command Line Interface Formulation Framework (cliff). The CLI is used to configure and deploy Oracle OpenStack for Oracle Linux using the Kolla OpenStack project, and Docker containers. This chapter gives you details on concepts used with the CLI. For a complete syntax reference for the CLI commands, see Appendix A, *Oracle OpenStack for Oracle Linux Toolkit CLI Reference*.

4.1 Using Deployment Groups

Deployment groups can be used for common deployment scenarios, such as for a set of compute nodes, controller nodes, storage nodes, and so on. A deployment group can be configured with one or more hosts and one or more services. There are a number of groups set up by default, which you may use, edit, or delete, according to your needs. The default groups are:

- **control**: Contains the control related services, such as glance, keystone, ndbcluster, nova, rabbitmq, and so on.
- **compute**: Contains the compute services, such as nova-compute.
- **database**: Contains the database related services.
- **network**: Contains the network related services, such as haproxy, neutron, and so on.
- **storage**: Contains the storage services, such as cinder.

To see a list of the deployment groups, and associated services, use the group listservices command.

Use the commands prefixed with `group` to manage deployment groups. See Section A.1, “kollacli” for a list of the commands you can use to manage deployment groups.

4.2 Using Ansible Configuration Properties

Ansible configuration files are used during the CLI deployment to install Oracle OpenStack for Oracle Linux. Use the commands prefixed with `property` to manage Ansible properties. When you set or clear the Ansible properties, the changes are written to the `/etc/kolla/globals.yml` file. The properties in the `globals.yml` file override any properties set in other files. The property files which are overridden are:

- `/etc/kolla/globals.yml`
- `/etc/kolla/passwords.yml`
- *`KOLLA_HOME`*`/group_vars/all.yml`
- *`KOLLA_HOME`*`/ansible/roles/`*`service_name`*`/default/main.yml`

Do not edit these files directly. Use the CLI to make changes to these files. To see a list of the Ansible properties and their associated values use the property list command. See Section A.1, “kollacli” for a list of the commands you can use to manage Ansible properties.

4.3 CLI Command Auto-Completion

Command auto-completion is available when using the CLI shell. You can set up CLI command auto-completion in your operating system using the `kollacli command` command. This command generates a script you can save to the operating system.

To set up CLI command auto-completion:

1. Install the bash-completion package, if it is not already installed:

   ```
   $ sudo yum install bash-completion
   ```

2. Run the `kollacli complete` command to generate the command auto-completion script. You can either send the output to a file:

   ```
   # kollacli complete >/etc/bash_completion.d/kollacli
   ```

   Or display it and copy/paste into a file:

   ```
   $ kollacli complete
   ```

   The script is displayed, for example:

   ```
   _kollacli()
   {
     local cur prev words
     COMPREPLY=()
     _get_comp_words_by_ref -n : cur prev words

     # Command data:
     cmds='complete deploy dump group help host list password property service setdeploy'
     cmds_complete='-h --help --name --shell'
     cmds_deploy='-h --help'
     cmds_dump='-h --help'
     cmds_group='add addhost listhosts listservices remove removehost'
     cmds_group_add='-h --help'
     cmds_group_addhost='-h --help'
     cmds_group_listhosts='-h --help -f --format -c --column --max-width --quote'
     cmds_group_listservices='-h --help -f --format -c --column --max-width --quote'
     cmds_group_remove='-h --help'
     cmds_group_removehost='-h --help'
     cmds_help='-h --help'
     cmds_host='add check destroy list remove setup'
     cmds_host_add='-h --help'
     cmds_host_check='-h --help'
     cmds_host_destroy='-h --help'
     cmds_host_list='-h --help -f --format -c --column --max-width --quote'
     cmds_host_remove='-h --help'
     cmds_host_setup='-h --help --insecure'
     cmds_password='clear list set'
     cmds_password_clear='-h --help'
     cmds_password_list='-h --help -f --format -c --column --max-width --quote'
     cmds_password_set='-h --help --insecure'
     cmds_property='clear list set'
     cmds_property_clear='-h --help'
     cmds_property_list='-h --help -f --format -c --column --max-width --quote'
     cmds_property_set='-h --help'
     cmds_service='addgroup list listgroups removegroup'
     cmds_service_addgroup='-h --help'
     cmds_service_list='-h --help -f --format -c --column --max-width --quote'
   ```

```
  cmds_service_listgroups='-h --help -f --format -c --column --max-width --quote'
  cmds_service_removegroup='-h --help'
  cmds_setdeploy='-h --help'

  cmd=""
  words[0]=""
  completed="${cmds}"
  for var in "${words[@]:1}"
  do
    if [[ ${var} == -* ]] ; then
      break
    fi
    if [ -z "${cmd}" ] ; then
      proposed="${var}"
    else
      proposed="${cmd}_${var}"
    fi
    local i="cmds_${proposed}"
    local comp="${!i}"
    if [ -z "${comp}" ] ; then
      break
    fi
    if [[ ${comp} == -* ]] ; then
      if [[ ${cur} != -* ]] ; then
        completed=""
        break
      fi
    fi
    cmd="${proposed}"
    completed="${comp}"
  done

  if [ -z "${completed}" ] ; then
    COMPREPLY=( $( compgen -f -- "$cur" ) $( compgen -d -- "$cur" ) )
  else
    COMPREPLY=( $(compgen -W "${completed}" -- ${cur}) )
  fi
  return 0
}
complete -F _kollacli kollacli
```

Copy and paste the output to a file, for example a file named `/etc/bash_completion.d/kollacli`. Writing to the `/etc/bash_completion.d` directory requires super user access (as root or using sudo).

3. Source the file to enable CLI command auto-completion:

```
$ source /etc/bash_completion.d/kollacli
```

4.4 Using the CLI Shell

The user which runs the CLI shell should be a member of the kolla group. To enter the CLI shell, enter:

```
$ kollacli
```

The CLI shell prompt is displayed `(kollacli)`. From the shell prompt you can enter CLI commands without using the `kollacli` command, for example:

```
(kollacli) host list
```

You can use the commands listed in this chapter using either the full syntax, or the CLI shell. The syntax shown in this appendix does not list the `kollacli` command. If you want to use the full command from the operating system prompt, make sure you enter `kollacli` before each command, for example:

```
$ kollacli host list
```

4.5 Getting Help with CLI Commands

There are multiple ways to get help when using the CLI commands. This section gives you examples on how to get help using the operating system shell, and the CLI shell.

To see a list of all CLI commands, use the `help` command. From the CLI shell, use:

```
(kollacli) help

Shell commands (type help <topic>):
===================================
cmdenvironment  edit  hi       l   list  pause  r    save  shell      show
ed              help  history  li  load  py     run  set   shortcuts

Undocumented commands:
======================
EOF  eof  exit  q  quit

Application commands (type help <topic>):
=========================================
complete            group remove      host remove     property set
deploy              group removehost  host setup      service addgroup
dump                help              password clear  service list
group add           host add          password list   service listgroups
group addhost       host check        password set    service removegroup
group listhosts     host destroy      property clear  setdeploy
group listservices  host list         property list
```

The CLI help, together with this documentation, cover the (cliff) shell and (kollacli) application commands listed in the output. You can get help on the cliff shell commands using the same syntax, for example:

```
(kollacli) help list
list [arg]: lists last command issued

        no arg -> list most recent command
        arg is integer -> list one history item, by index
        a..b, a:b, a:, ..b -> list spans from a (or start) to b (or end)
        arg is string -> list all commands matching string search
        arg is /enclosed in forward-slashes/ -> regular expression search
```

If you are using the operating system shell, the output is slightly different when you use the `help` command, but still displays a list of the commands to help you.

```
$ kollacli help
usage: kollacli [--version] [-v] [--log-file LOG_FILE] [-q] [-h] [--debug]

Command-Line Client for OpenStack Kolla

optional arguments:
  --version            show program's version number and exit
  -v, --verbose        Increase verbosity of output. Can be repeated.
  --log-file LOG_FILE  Specify a file to log output. Disabled by default.
  -q, --quiet          suppress output except warnings and errors
  -h, --help           show this help message and exit
  --debug              show tracebacks on errors

Commands:
  complete       print bash completion command
  deploy         Deploy
  dump           Dumps configuration data for debugging
  group add      Add group to open-stack-kolla
  group addhost  Add host to group
  group listhosts  List all groups and their hosts
  group listservices  List all groups and their services
  group remove   Remove group from openstack-kolla
  group removehost  Remove host group from group
  help           print detailed help for another command
  host add       Add host to open-stack-kolla
  host check     Check if openstack-kollacli is setup
  host destroy   Destroy
  host list      List hosts and their groups
  host remove    Remove host from openstack-kolla
  host setup     Setup openstack-kollacli on host
  password clear  Password Clear
  password list  List all password names
  password set   Password Set
  property clear  Property Clear
  property list  List all properties
  property set   Property Set
  service addgroup  Add group to service
  service list   List services and their sub-services
  service listgroups  List services and their groups
  service removegroup  Remove group from service
  setdeploy      Set deploy mode
```

You can get help on a specific command using the `help` command, and passing in the command name. For example, to get a list of all the host related commands, use:

```
(kollacli) help host
```

Or from the operating system shell:

```
$ kollacli help host
```

Which displays a list of all the host commands:

```
Command "host" matches:
  host add
  host check
  host destroy
  host list
  host remove
  host setup
```

Each CLI command has a `-h` or `--help` option, which displays syntax and usage for the command. You can also use the `help` command, followed by the command name to achieve the same result. For example, the following commands all display the same output:

```
(kollacli) host list -h
(kollacli) host list --help
(kollacli) help host list
```

The output of all these commands is:

```
usage: host list [-h] [-f {csv,html,json,table,value,yaml}] [-c COLUMN]
                 [--max-width <integer>]
                 [--quote {all,minimal,none,nonnumeric}]
                 [[hostname]]

List hosts and their groups If a hostname is provided, only list information
about that host.

positional arguments:
  [hostname]            hostname

optional arguments:
  -h, --help            show this help message and exit
```

```
output formatters:
  output formatter options

  -f {csv,html,json,table,value,yaml}, --format {csv,html,json,table,value,yaml}
                        the output format, defaults to table
  -c COLUMN, --column COLUMN
                        specify the column(s) to include, can be repeated

table formatter:
  --max-width <integer>
                        Maximum display width, 0 to disable

CSV Formatter:
  --quote {all,minimal,none,nonnumeric}
                        when to include quotes, defaults to nonnumeric
```

Chapter 5 Troubleshooting

Table of Contents

This chapter contains some troubleshooting tips to help you with your deployment.

5.1 Removing OpenStack Services from Hosts

You may want to kill and remove any OpenStack services that are installed and running on a target node. For example, if a deployment fails on a target node, and you want to clean the host before you redeploy. To do this, you can use the `kollacli host destroy` command. This command kills and removes all OpenStack services (Docker containers) on a host:

```
$ kollacli host destroy mycomputenode1.example.com
```

Or on all hosts:

```
$ kollacli host destroy all
```

5.2 Removing Images from Local Copy of Docker Registry

You may want to remove all images from the local copy of the Docker registry. To remove all images, run:

```
$ sudo docker rmi -f $(docker images -q)
```

5.3 Collecting the Log Files

The `log_collector.py` script enables you collect the Docker log files from the containers in your OpenStack deployment. The script is included in the `openstack-kollacli` package, which is installed on *master* nodes, and is located in `/usr/share/kolla/kollacli/tools`. You collect the log files from all nodes, or from selected nodes, as follows:

`log_collector.py` [`all` | *`host1`* [`,`*`host2`*`,`*`host3`*`...`]] [`-h` | `--help`]

If you specify a host, the host must have been added to the deployment inventory using the `kollacli host add` command. If a host is not accessible, the script ignores the host and processes the next host in the list.

The script only collects logs from Docker containers whose image names start with "ol-openstack".

The script collects the logs files into a tar archive in `/tmp` and prints the name of the file to the screen. The archive contains a directory for each specified deployment node, and each directory contains the Docker log files for the containers deployed on that node.

The script also runs the `kollacli dump` command, which retrieves the Kolla configuration files and logs, and includes the output of this command in the archive.

Appendix A Oracle OpenStack for Oracle Linux Toolkit CLI Reference

Table of Contents

This appendix contains information on the Oracle OpenStack for Oracle Linux toolkit CLI syntax and usage. The toolkit command is started using the `kollacli` command.

A.1 kollacli

Starts the CLI shell, or runs CLI commands from the operating system prompt.

Syntax

`kollacli` [*`command`*] [`--version`] [`-v` | `--verbose`] [`--log-file` *`file_name`*] [`-q` | `--quiet`] [`-h` | `--help`] [`--debug`]

Description

This command starts the CLI shell, or runs CLI commands from the operating system prompt. A list of the commands you can enter are in Section A.1.1, “kollacli Command Options”.

Options

The following table shows the available options for this command.

Option	Description
`command`	A `kollacli` sub-command. The sub-commands are listed in Section A.1.1, “kollacli Command Options”.
`--version`	Displays the CLI version number.
[`-v` | `--verbose`]	Sets the verbosity of the output when running a command. Provides more output when you run a command, which varies for each command. This may be useful in helping you resolve errors, especially when using the `deploy` command. If you repeat `-v`, you increase the verbosity, so `-vv` gives you more than `-v`, and `-vvv` gives you more than `-vv`.
[`--log-file` *`file_name`*]	Set the file name to log debug and error output. This is disabled by default. The output is in verbose format. If the log file exists, the results are appended to the existing file content.
[`-q` | `--quiet`]	Suppress all output, except warnings and errors.
[`-h` | `--help`]	Displays the CLI syntax help.
`--debug`	Displays traceback on errors.

Examples

Example A.1 Starting the CLI shell

```
$ kollacli
```

Example A.2 Running a sub-command from the operating system prompt

```
$ kollacli deploy
```

Example A.3 Showing the CLI version number

```
$ kollacli --version
```

Example A.4 Running a deployment with increased output

```
$ kollacli deploy -v
```

Example A.5 Running a deployment, and save output to a file while suppressing screen output

```
$ kollacli deploy --log-file log-out.txt --quiet --debug
```

A.1.1 kollacli Command Options

This section contains the complete syntax reference, with examples, for each `kollacli` sub-command. These commands are used with the *`commands`* option of the `kollacli` command.

A.1.1.1 complete

Provides a script to use for auto-completion of CLI commands in the operating system shell.

Syntax

`complete` [`--name` *`function_name`*] [`--shell` { `bash` | `none` }] [`-h` | `--help`]

Description

This command generates a script to use for CLI command auto-completion when typing CLI commands into the operating system shell. If you supply a function name, the generated script is formatted as a function with the name you provide. Copy the output and save it to a file. Source the script to enable CLI command auto-completion from your shell. For example, if you create a function name of `kollacliCommands`, use:

```
$ source kollacliCommands
```

Using this command without any options prints the default script with a function name of `kollacli`.

Options

The following table shows the available options for this command.

Option	Description
`--name` *`function_name`*	The name of the function to generate. The default function name is `kollacli`.
`--shell` { `bash` | `none` }	The shell type to use for the script. Use `none` for data only. The default is `bash`.

Option	Description
[`-h` | `--help`]	Displays help on syntax and usage for this command.

Examples

Example A.6 Creating a script for operating system command auto-completion

Using the shell:

```
(kollacli) complete --name "kollacliCommands"
```

Using the operating system command prompt:

```
$ kollacli complete --name "kollacliCommands"
```

A.1.1.2 deploy

Performs the deployment of Oracle OpenStack for Oracle Linux on all the configured hosts.

Syntax

`deploy` [`--serial`] [`--services` *`service_name`* ...] [`--groups` *`group_name`* ... | `--hosts` *`hostname`* ...] [`-h` | `--help`]

Description

This command performs the deployment of Oracle OpenStack for Oracle Linux services on hosts. You can fine tune the deployment of the OpenStack services in a number of ways, by service, host, deployment group, or any combination of those. For example, you could deploy:

- All services, on all hosts
- Services on hosts in parallel, or serially
- One or more services to one or more hosts
- One or more services to one or more deployment groups

Your deployment model may depend on whether you are deploying an environment for the first time, or updating OpenStack services:

- For an initial deployment, you may choose to deploy all configured services to all hosts, in parallel.
- For an update of the services, you must use only the `--serial` option. The data containers used by the OpenStack services are preserved when you update.

Options

The following table shows the available options for this command.

Option	Description
`--serial`	Sets the deployment to be performed serially (one host at a time). Serial deployment is useful for updating existing OpenStack services. Serial deployment in an HA environment, means you can update a deployment with no service interruptions.

Option	Description
`--services` *`service_name`* ...	Sets the list of services to be deployed. To get a list of available services, use the service list command. Use a comma (`,`) to separate services in a list, for example: `--services mysqlcluster,keystone,neutron`
[`--groups` *`group_name`* ... | `--hosts` *`hostname`* ...]	You can specify one or more deployment groups, for example, a group of hosts in an HA group, or all controller nodes. To get a list of available deployment groups, and the hosts in those groups, use the group listhosts command. Alternatively, you can specify one or more hosts. To get a list of available hosts, and the deployment groups with which the hosts are associated, use the host list command. Use a comma (`,`) to separate groups or hosts in a list, for example: `--groups compute,control,storage`
[`-h` | `--help`]	Displays help on syntax and usage for this command.

Examples

Example A.7 Deploying all services to all hosts

Using the shell:

```
(kollacli) deploy
```

Using the operating system command prompt:

```
$ kollacli deploy
```

Example A.8 Deploying all services serially to a list of deployment groups

Using the shell:

```
(kollacli) deploy --serial --groups myHAGroup1,myHAGroup2
```

Using the operating system command prompt:

```
$ kollacli deploy --serial --groups myHAGroup1,myHAGroup2
```

Example A.9 Deploying all services serially to a list of hosts

Using the shell:

```
(kollacli) deploy --serial --hosts mycomputenode1.example.com,mycomputenode2.example.com
```

Using the operating system command prompt:

```
$ kollacli deploy --serial --hosts mycomputenode1.example.com,mycomputenode2.example.com
```

Example A.10 Deploying a list of services to a deployment group

Using the shell:

```
(kollacli) deploy --services mysqlcluster,neutron,keystone --groups control
```

Using the operating system command prompt:

```
$ kollacli deploy --services mysqlcluster,neutron,keystone --groups control
```

A.1.1.3 dump

Creates a tar file of the Kolla configuration and log files for debugging Oracle OpenStack for Oracle Linux issues.

Syntax

`dump` [`-h` | `--help`]

Description

This command copies the Kolla configuration files log files for Oracle OpenStack for Oracle Linux, and saves them to a tar file. These files may be useful when speaking to Oracle about any problems you are facing. The files included in the tar file are:

- `/etc/kolla/*`
- `/usr/share/kolla/*`

The tar file is saved to the `/tmp` directory, and the name of the file printed to the screen.

Options

The following table shows the available options for this command.

Option	Description
[`-h` | `--help`]	Displays help on syntax and usage for this command.

Examples

Example A.11 Creating a compressed file of log files

Using the shell:

```
(kollacli) dump
```

Using the operating system command prompt:

```
$ kollacli dump
```

A.1.1.4 exit

Quits and exits the CLI.

Syntax

`exit`

Description

This command exists the CLI and returns you to the operating system prompt.

This option is not available from the operating system command prompt

Options

There are no options for this command.

Examples

Example A.12 Exiting the CLI

Using the shell:

```
(kollacli) exit
```

A.1.1.5 group add

Adds a deployment group.

Syntax

`group add` *`group_name`* [`-h` | `--help`]

Description

This command adds a deployment group. There are five deployment groups set up by default: compute, control, database, network, and storage. For information on using deployment groups, see Section 4.1, “Using Deployment Groups”.

Options

The following table shows the available options for this command.

Option	Description
`group_name`	The name of the deployment group.
[`-h` | `--help`]	Displays help on syntax and usage for this command.

Examples

Example A.13 Creating a group

Using the shell:

```
(kollacli) group add mygroup
```

Using the operating system command prompt:

```
$ kollacli group add mygroup
```

A.1.1.6 group addhost

Adds a host to a deployment group.

Syntax

`group addhost` *`group_name`* *`host`* [`-h` | `--help`]

Description

This command adds a host to a deployment group.

Options

The following table shows the available options for this command.

Option	Description
`group_name`	The name of the deployment group. If the deployment group does not yet exist, use the group add command to create it.
`host`	The fully qualified domain name or IP address of the host.
[`-h` | `--help`]	Displays help on syntax and usage for this command.

Examples

Example A.14 Adding a host to a deployment group

Using the shell:

```
(kollacli) group addhost mygroup myhost.example.com
```

Using the operating system command prompt:

```
$ kollacli group addhost mygroup myhost.example.com
```

A.1.1.7 group listhosts

Lists the deployment groups and the hosts included in those groups.

Syntax

`group listhosts` [{ `-f` | `--format` } *`file_options`*] [{ `-c` | `--column` } *`column_name`* ...] [`--max-width` *`max_width`*] [`--quote` *`quote_options`*] [`-h` | `--help`]

Where *`file_options`* is:

{ `csv` | `html` | `json` | `table` | `value` | `yaml` }

And *`quote_options`* is:

{ `all` | `minimal` | `none` | `nonnumeric` }

Description

This command lists all deployment groups, and the hosts included in each group.

Options

The following table shows the available options for this command.

Option	Description
{ `-f` | `--format` } *`file_options`*	Allows you to format the output in various formats. The *`file_options`* option may be one of: • `csv`: Comma Separated Values.

Option	Description
	• `html`: HTML `table` markup. • `json`: JavaScript Object Notation. • `table`: Simple ASCII display table. • `value`: Space separated values with no headers. This format may be useful to pipe output to an operating system command. • `yaml`: YAML format is used natively by Ansible playbooks. The default format is `table`.
{ `-c` | `--column` } *`column_name`* ...	Allows you to specify which column(s) to lists. The *`column_name`* option may be either `Group`, or `Hosts` (column names are case sensitive). You can repeat this option to include multiple columns. The default is to include all columns.
`--max-width` *`max_width`*	Sets the maximum display width for each column when using `table` output (`table` is the default output). The *`max_width`* must be an integer. The default for *`max_width`* is `0` (no maximum width).
`--quote` *`quote_options`*	Sets the quoting style for when using *`csv`* output. The *`quote_options`* option may be one of: • `all`: Quote all values. • `minimal`: Optimized minimal quoting of values. • `none`: No quoting of any values. • `nonnumeric`: Quote only non-numeric values. The default is `nonnumeric`.
[`-h` | `--help`]	Displays help on syntax and usage for this command.

Examples

Example A.15 Listing the deployment groups and associated hosts

Using the shell:

```
(kollacli) group listhosts
```

Using the operating system command prompt:

```
$ kollacli group listhosts
```

Example A.16 Listing the deployment groups and associated hosts in YAML format

Using the shell:

```
(kollacli) group listhosts -f yaml
```

Using the operating system command prompt:

```
$ kollacli group listhosts -f yaml
```

Example A.17 Listing the deployment groups in CSV format with formatting options

Using the shell:

```
(kollacli) group listhosts -f csv -c Group --quote minimal
```

Using the operating system command prompt:

```
$ kollacli group listhosts -f csv -c Group --quote minimal
```

Example A.18 Listing the deployment groups and associated hosts in table format with formatting options

Using the shell:

```
(kollacli) group listhosts --max-width 20
```

Using the operating system command prompt:

```
$ kollacli group listhosts --max-width 20
```

A.1.1.8 group listservices

Lists the deployment groups and the OpenStack services included in those groups.

Syntax

`group listservices` [{ `-f` | `--format` } *`file_options`*] [{ `-c` | `--column` } *`column_name`* ...] [`--max-width` *`max_width`*] [`--quote` *`quote_options`*] [`-h` | `--help`]

Where *`file_options`* is:

{ `csv` | `html` | `json` | `table` | `value` | `yaml` }

And *`quote_options`* is:

{ `all` | `minimal` | `none` | `nonnumeric` }

Description

This command lists all deployment groups, and the OpenStack services included in each group.

Options

The following table shows the available options for this command.

Option	Description
{ `-f` | `--format` } *`file_options`*	Allows you to format the output in various formats. The *`file_options`* option may be one of: • `csv`: Comma Separated Values. • `html`: HTML `table` markup. • `json`: JavaScript Object Notation. • `table`: Simple ASCII display table.

Option	Description
	• `value`: Space separated values with no headers. This format may be useful to pipe output to an operating system command. • `yaml`: YAML format is used natively by Ansible playbooks. The default format is `table`.
{ `-c` | `--column` } *`column_name`* ...	Allows you to specify which column(s) to lists. The *`column_name`* option may be either `Group`, or `Services` (column names are case sensitive). You can repeat this option to include multiple columns. The default is to include all columns.
`--max-width` *`max_width`*	Sets the maximum display width for each column when using `table` output (`table` is the default output). The *`max_width`* must be an integer. The default for *`max_width`* is `0` (no maximum width).
`--quote` *`quote_options`*	Sets the quoting style for when using *`csv`* output. The *`quote_options`* option may be one of: • `all`: Quote all values. • `minimal`: Optimized minimal quoting of values. • `none`: No quoting of any values. • `nonnumeric`: Quote only non-numeric values. The default is `nonnumeric`.
[`-h` | `--help`]	Displays help on syntax and usage for this command.

Examples

Example A.19 Listing the deployment groups and associated services

Using the shell:

```
(kollacli) group listservices
```

Using the operating system command prompt:

```
$ kollacli group listservices
```

Example A.20 Listing the deployment groups and associated services in YAML format

Using the shell:

```
(kollacli) group listservices -f yaml
```

Using the operating system command prompt:

```
$ kollacli group listservices -f yaml
```

Example A.21 Listing the deployment groups in CSV format with formatting options

Using the shell:

```
(kollacli) group listservices -f csv -c Group --quote minimal
```

Using the operating system command prompt:

```
$ kollacli group listservices -f csv -c Group --quote minimal
```

Example A.22 Listing the deployment groups and associated services in table format with formatting options

Using the shell:

```
(kollacli) group listservices --max-width 60
```

Using the operating system command prompt:

```
$ kollacli group listservices --max-width 60
```

A.1.1.9 group remove

Removes a deployment group.

Syntax

`group remove` *`group_name`* [`-h` | `--help`]

Description

This command removes a deployment group.

Options

The following table shows the available options for this command.

Option	Description
`group_name`	The name of the deployment group.
[`-h` | `--help`]	Displays help on syntax and usage for this command.

Examples

Example A.23 Removing a group

Using the shell:

```
(kollacli) group remove mygroup
```

Using the operating system command prompt:

```
$ kollacli group remove mygroup
```

A.1.1.10 group removehost

Removes a host from a deployment group.

Syntax

`group removehost` *`group_name`* *`host`* [`-h` | `--help`]

Description

This command removes a host from a deployment group.

Options

The following table shows the available options for this command.

Option	Description
group_name	The name of the deployment group. If the deployment group does not yet exist, use the group add command to create it.
host	The fully qualified domain name or IP address of the host.
[-h | --help]	Displays help on syntax and usage for this command.

Examples

Example A.24 Removing a host from a deployment group

Using the shell:

```
(kollacli) group removehost mygroup myhost.example.com
```

Using the operating system command prompt:

```
$ kollacli group removehost mygroup myhost.example.com
```

A.1.1.11 help

Displays help on using CLI commands.

Syntax

help [*command*] [-h | --help]

Description

This command displays help on using CLI commands. Using this command without any options displays a list of all CLI commands. For an overview of getting help when using the CLI, see Section 4.5, “Getting Help with CLI Commands”.

Options

The following table shows the available options for this command.

Option	Description
command	The name of the command for which to display help.
[-h | --help]	Displays help on syntax and usage for this command.

Examples

Example A.25 Adding a host

Using the shell:

```
(kollacli) help "service listgroups"
```

Using the operating system command prompt:

```
$ kollacli help "service listgroups"
```

A.1.1.12 host add

Adds a host.

Syntax

`host add` *`host`* [`-h` | `--help`]

Description

This command adds a host to be included as a node in the deployment of Oracle OpenStack for Oracle Linux. After adding a host, you should:

1. Optionally, distribute the SSH keys and perform pre-deployment checks using the host setup command.
2. Add the host to a deployment group using the group addhost command.

Options

The following table shows the available options for this command.

Option	Description
`host`	The fully qualified domain name or IP address of the host.
[`-h` | `--help`]	Displays help on syntax and usage for this command.

Examples

Example A.26 Adding a host

Using the shell:

```
(kollacli) host add myhost.example.com
```

Using the operating system command prompt:

```
$ kollacli host add myhost.example.com
```

A.1.1.13 host check

Checks a host.

Syntax

`host check` *`host`* [`-h` | `--help`]

Description

This command verifies the host is in good working order. Performs both the pre- and post-installation checks described in the host setup command. The host must first be added using the host add command.

Options

The following table shows the available options for this command.

Option	Description
host	The fully qualified domain name or IP address of the host.
[-h | --help]	Displays help on syntax and usage for this command.

Examples

Example A.27 Checking a host's configuration

Using the shell:

```
(kollacli) host check myhost.example.com
```

Using the operating system command prompt:

```
$ kollacli host check myhost.example.com
```

A.1.1.14 host destroy

Stops and removes Kolla-related Docker containers on one or all hosts.

Syntax

`host destroy` { *host* | `all` } [`--stop`] [`--includedata`] [`-h` | `--help`]

Description

This command stops and removes Kolla-related Docker containers on the specified host, or from all hosts. By default, the containers are killed (`docker kill`) before they are removed and OpenStack data containers are not removed.

Options

The following table shows the available options for this command.

Option	Description
{ *host* | all }	The fully qualified domain name or IP address of the host. Alternatively, you can use `all` to destroy the containers on all hosts.
--stop	Performs a graceful shutdown (`docker stop`) of the containers before they are removed. Using this option might increase the time needed for the command to complete, but it ensures that the containers are removed cleanly.
--includedata	Removes the OpenStack data containers from the host.
[-h | --help]	Displays help on syntax and usage for this command.

Examples

Example A.28 Killing and removing Kolla containers on all hosts

Using the shell:

```
(kollacli) host destroy all
```

Using the operating system command prompt:

```
$ kollacli host destroy all
```

Example A.29 Killing and removing all Kolla containers on a host

Using the shell:

```
(kollacli) host destroy --includedata mycontrolnode.example.com
```

Using the operating system command prompt:

```
$ kollacli host destroy --includedata mycontrolnode.example.com
```

A.1.1.15 host list

Lists the hosts and the deployment groups with which the hosts are associated.

Syntax

`host list` [*`host`*] [{ `-f` | `--format` } *`file_options`*] [{ `-c` | `--column` } *`column_name`* ...] [`--max-width` *`max_width`*] [`--quote` *`quote_options`*] [`-h` | `--help`]

Where *`file_options`* is:

{ `csv` | `html` | `json` | `table` | `value` | `yaml` }

And *`quote_options`* is:

{ `all` | `minimal` | `none` | `nonnumeric` }

Description

This command lists the hosts and the deployment groups with which the hosts are associated.

Options

The following table shows the available options for this command.

Option	Description
`host`	The fully qualified domain name or IP address of the host. If no *`host`* is provided, all hosts are listed.
{ `-f` | `--format` } *`file_options`*	Allows you to format the output in various formats. The *`file_options`* option may be one of: • `csv`: Comma Separated Values. • `html`: HTML `table` markup. • `json`: JavaScript Object Notation. • `table`: Simple ASCII display table. • `value`: Space separated values with no headers. This format may be useful to pipe output to an operating system command.

Option	Description
	• `yaml`: YAML format is used natively by Ansible playbooks. The default format is `table`.
{ `-c` | `--column` } *`column_name`* ...	Allows you to specify which column(s) to lists. The *`column_name`* option may be either `Groups`, or `Hosts` (column names are case sensitive). You can repeat this option to include multiple columns. The default is to include all columns.
`--max-width` *`max_width`*	Sets the maximum display width for each column when using `table` output (`table` is the default output). The *`max_width`* must be an integer. The default for *`max_width`* is `0` (no maximum width).
`--quote` *`quote_options`*	Sets the quoting style for when using *`csv`* output. The *`quote_options`* option may be one of: • `all`: Quote all values. • `minimal`: Optimized minimal quoting of values. • `none`: No quoting of any values. • `nonnumeric`: Quote only non-numeric values. The default is `nonnumeric`.
[`-h` | `--help`]	Displays help on syntax and usage for this command.

Examples

Example A.30 Listing the hosts and associated deployment groups

Using the shell:

```
(kollacli) host list
```

Using the operating system command prompt:

```
$ kollacli host list
```

Example A.31 Listing the hosts and associated deployment groups in YAML format

Using the shell:

```
(kollacli) host list -f yaml
```

Using the operating system command prompt:

```
$ kollacli host list -f yaml
```

Example A.32 Listing the deployment groups for a host in CSV format with formatting options

Using the shell:

```
(kollacli) host list myhost.example.com -f csv -c Groups --quote minimal
```

Using the operating system command prompt:

```
$ kollacli host list myhost.example.com -f csv -c Groups --quote minimal
```

Example A.33 Listing the deployment groups for a host in table format with formatting options

Using the shell:

```
(kollacli) host list myhost.example.com --max-width 20
```

Using the operating system command prompt:

```
$ kollacli host list myhost.example.com --max-width 20
```

A.1.1.16 host remove

Removes a host.

Syntax

`host remove` *`host`* [`-h` | `--help`]

Description

This command removes a host from the list of target nodes.

Options

The following table shows the available options for this command.

Option	Description
`host`	The fully qualified domain name or IP address of the host.
[`-h` | `--help`]	Displays help on syntax and usage for this command.

Examples

Example A.34 Removing a host

Using the shell:

```
(kollacli) host remove myhost.example.com
```

Using the operating system command prompt:

```
$ kollacli host remove myhost.example.com
```

A.1.1.17 host setup

Distributes SSH keys and confirms login to the host.

Syntax

`host setup` { *`host`* | { `-f` | `--file` } *`file_name`* } [`-h` | `--help`]

Description

This command copies the SSH public key for the kolla user from the master nodes to the specified hosts and then checks that the kolla user can log in to the hosts using SSH keys. The SSH keys are created for the kolla user when the Oracle OpenStack for Oracle Linux preinstall package (openstack-kolla-preinstall) is installed on the master node.

The hosts must already be added to the OpenStack deployment with the `kollacli host add` command, and the Docker service must be installed and running on the hosts.

You specify a single *`host`* to set up, or use the `-f` *`file_name`* option to specify a file that contains the details of one or more hosts to set up.

The command requires a user name and password to connect to the target host. By default, the root user is used. You can specify a different user in the `KOLLA_CLI_SETUP_USER` environment variable, or use the `-f` option and specify a user name in a file. The user must already exist on the target host and have sufficient privileges to be able to write to the `/usr/share/kolla/.ssh/authorized_keys` file, which is owned by the kolla user and group. When you specify a *`host`*, the command prompts you for the password. If you use the `-f` option, the file contains the passwords to use for each host.

Caution

If you use the `-f` option, make sure you secure the file containing the login credentials.

Options

The following table shows the available options for this command.

Option	Description
`host`	The fully qualified domain name or IP address of the host.

{ `-f` | `--file` } *`file_name`*	Specifies the absolute path of a YAML format file that contains the names of the hosts to set up and the passwords to use. If the file specifies a user name for a host (optional), this is used instead of root or the user name in the `KOLLA_CLI_SETUP_USER` environment variable. An example YAML file is:
	`---`
	`mycontrolnode1.example.com:`
	`    password: password`
	`mycontrolnode2.example.com:`
	`    password: password`
	`mycomputenode1.example.com:`
	`    password: password`
	`    uname: user_name`
	`mycomputenode2.example.com:`
	`    password: password`
	`    uname: user_name`
	`mycomputenode3.example.com:`
	`    password: password`
	`    uname: user_name`
	`mycomputenode4.example.com:`
	`    password: password`
	`    uname: user_name`
	`mystoragenode1.example.com:`
	`    password: password`
	`mystoragenode2.example.com:`
	`    password: password`

[`-h` | `--help`]	Displays help on syntax and usage for this command.

Examples

Example A.35 Setting up a host

Using the shell:

```
(kollacli) host setup myhost.example.com
```

Using the operating system command prompt:

```
$ kollacli host setup myhost.example.com
```

Example A.36 Setting up multiple hosts

Using the shell:

```
(kollacli) host setup --file /myhome/hosts.yml
```

Using the operating system command prompt:

```
$ kollacli host setup --file /myhome/hosts.yml
```

A.1.1.18 password clear

Removes a password name and value from the passwords file.

Syntax

`password clear` *`password_name`* [`-h` | `--help`]

Description

This command removes a password and value from the `/etc/kolla/passwords.yml` file.

Important

The `/etc/kolla/passwords.yml` file is only visible to the root or kolla user. Do not edit this file directly; you must use the related `password` commands to make changes to the passwords file.

Options

The following table shows the available options for this command.

Option	Description
`password_name`	The name of the password. To see a list of the password names, use the password list command.
[`-h` | `--help`]	Displays help on syntax and usage for this command.

Examples

Example A.37 Removing a password and value

Using the shell:

```
(kollacli) password clear database_password
```

Using the operating system command prompt:

```
$ kollacli password clear database_password
```

A.1.1.19 password list

Lists the password names.

Syntax

`password list` [`-h` | `--help`]

Description

This command lists all password names contained in the `/etc/kolla/passwords.yml` file. Values for the passwords are not displayed.

Options

The following table shows the available options for this command.

Option	Description
[`-h` | `--help`]	Displays help on syntax and usage for this command.

Examples

Example A.38 Listing password names

Using the shell:

```
(kollacli) password list
```

Using the operating system command prompt:

```
$ kollacli password list
```

A.1.1.20 password set

Sets a password to be used during deployment.

Syntax

`password set` *`password_name`* [`-h` | `--help`]

Description

This command sets a password value to be used during the deployment. The passwords are stored in the `/etc/kolla/passwords.yml` file.

Important

The `/etc/kolla/passwords.yml` file is only visible to the root or kolla user. Do not edit this file directly; you must use the related `password` commands to make changes to the passwords file.

You are prompted to enter the password value when the command is executed. The password is not displayed on screen. If a password name does not exist in the `passwords.yml` file, this command creates it. If a password name does exist, the password value is replaced with the new value. In a production environment, all passwords should be set to make sure your environment is secure.

Options

The following table shows the available options for this command.

Option	Description
password_name	The name of the password. To see a list of the password names, use the password list command.
[-h | --help]	Displays help on syntax and usage for this command.

Examples

Example A.39 Setting a password

Using the shell:

```
(kollacli) password set database_password
```

Using the operating system command prompt:

```
$ kollacli password set database_password
```

A.1.1.21 property clear

Clears a property.

Syntax

`property clear` *`property_name`* `[ -h | --help ]`

Description

This command clears a property value used to set configuration options in the Ansible configuration files.

Options

The following table shows the available options for this command.

Option	Description
property_name	The name of the Ansible property. To see a list of the Ansible properties, associated values, and the file to which they are written, use the property list command.
[-h | --help]	Displays help on syntax and usage for this command.

Examples

Example A.40 Clearing an Ansible property

Using the shell:

```
(kollacli) property clear enable_haproxy
```

Using the operating system command prompt:

```
$ kollacli property clear enable_haproxy
```

A.1.1.22 property list

Lists the Ansible properties and associated values.

Syntax

`property list` [{ `-f` | `--format` } *`file_options`*] [{ `-c` | `--column` } *`column_name`* ...] [`--max-width` *`max_width`*] [`--quote` *`quote_options`*] [`-h` | `--help`]

Where *`file_options`* is:

{ `csv` | `html` | `json` | `table` | `value` | `yaml` }

And *`quote_options`* is:

{ `all` | `minimal` | `none` | `nonnumeric` }

Description

This command lists the Ansible properties and associated values.

Options

The following table shows the available options for this command.

Option	Description
{ `-f` | `--format` } *`file_options`*	Allows you to format the output in various formats. The *`file_options`* option may be one of: • `csv`: Comma Separated Values. • `html`: HTML `table` markup. • `json`: JavaScript Object Notation. • `table`: Simple ASCII display table. • `value`: Space separated values with no headers. This format may be useful to pipe output to an operating system command. • `yaml`: YAML format is used natively by Ansible playbooks. The default format is `table`.
{ `-c` | `--column` } *`column_name`* ...	Allows you to specify which column(s) to lists. The *`column_name`* option may be `Property Name`, or `Property Value` (column names are case sensitive). You can repeat this option to include multiple columns. The default is to include all columns.
`--max-width` *`max_width`*	Sets the maximum display width for each column when using `table` output (`table` is the default output). The *`max_width`* must be an integer. The default for *`max_width`* is `0` (no maximum width).
`--quote` *`quote_options`*	Sets the quoting style for when using *`csv`* output. The *`quote_options`* option may be one of: • `all`: Quote all values. • `minimal`: Optimized minimal quoting of values.

Option	Description
	• `none`: No quoting of any values. • `nonnumeric`: Quote only non-numeric values. The default is `nonnumeric`.
[`-h` | `--help`]	Displays help on syntax and usage for this command.

Examples

Example A.41 Listing the properties and associated values

Using the shell:

```
(kollacli) property list
```

Using the operating system command prompt:

```
$ kollacli property list
```

Example A.42 Listing the properties and associated values in YAML format

Using the shell:

```
(kollacli) property list -f yaml
```

Using the operating system command prompt:

```
$ kollacli property list -f yaml
```

Example A.43 Listing the property names in CSV format with formatting options

Using the shell:

```
(kollacli) property list -f csv -c "Property Name" --quote minimal
```

Using the operating system command prompt:

```
$ kollacli property list -f csv -c "Property Name" --quote minimal
```

Example A.44 Listing the properties and associated values in table format with formatting options

Using the shell:

```
(kollacli) property list --max-width 20
```

Using the operating system command prompt:

```
$ kollacli property list --max-width 20
```

A.1.1.23 property set

Sets a property.

Syntax

`property set` *`property_name`* *`value`* [`-h` | `--help`]

Description

This command sets a property value used to set configuration options in the Ansible configuration files.

Options

The following table shows the available options for this command.

Option	Description
property_name	The name of the Ansible property. To see a list of the Ansible properties, associated values, and the file to which they are written, use the property list command.
value	The value to set for the property.
[-h | --help]	Displays help on syntax and usage for this command.

Examples

Example A.45 Setting an Ansible property

Using the shell:

```
(kollacli) property set enable_haproxy yes
```

Using the operating system command prompt:

```
$ kollacli property set enable_haproxy yes
```

A.1.1.24 quit

Quits and exits the CLI.

Syntax

`quit`

Description

This command exists the CLI and returns you to the operating system prompt.

This option is not available from the operating system command prompt

Options

There are no options for this command.

Examples

Example A.46 Exiting the CLI

Using the shell:

```
(kollacli) quit
```

A.1.1.25 service addgroup

Adds a deployment group to an OpenStack service.

Syntax

`service addgroup` *`service_name`* *`group_name`* [`-h` | `--help`]

Description

This command adds a deployment group to an OpenStack service.

Options

The following table shows the available options for this command.

Option	Description
service_name	The OpenStack service name. To show a list of available OpenStack services, and the groups to which they belong, use the service list command.
group_name	The name of the deployment group. If the deployment group does not yet exist, use the group add command to create it.
[-h | --help]	Displays help on syntax and usage for this command.

Examples

Example A.47 Adding a deployment group to a service

Using the shell:

```
(kollacli) service addgroup neutron mygroup
```

Using the operating system command prompt:

```
$ kollacli service addgroup neutron mygroup
```

A.1.1.26 service list

Lists all services and associated sub-services.

Syntax

service list [{ -f | --format } *file_options*] [{ -c | --column } *column_name* ...] [--max-width *max_width*] [--quote *quote_options*] [-h | --help]

Where *file_options* is:

{ csv | html | json | table | value | yaml }

And *quote_options* is:

{ all | minimal | none | nonnumeric }

Description

This command lists all the services and the associated sub-services. This command also shows services which are not part of any group.

Options

The following table shows the available options for this command.

Option	Description
{ -f | --format } *file_options*	Allows you to format the output in various formats. The *file_options* option may be one of:

Option	Description
	• `csv`: Comma Separated Values. • `html`: HTML `table` markup. • `json`: JavaScript Object Notation. • `table`: Simple ASCII display table. • `value`: Space separated values with no headers. This format may be useful to pipe output to an operating system command. • `yaml`: YAML format is used natively by Ansible playbooks. The default format is `table`.
{ `-c` | `--column` } *`column_name`* ...	Allows you to specify which column(s) to lists. The *`column_name`* option may be either `Service`, or `Groups` (column names are case sensitive). You can repeat this option to include multiple columns. The default is to include all columns.
`--max-width` *`max_width`*	Sets the maximum display width for each column when using `table` output (`table` is the default output). The *`max_width`* must be an integer. The default for *`max_width`* is `0` (no maximum width).
`--quote` *`quote_options`*	Sets the quoting style for when using *`csv`* output. The *`quote_options`* option may be one of: • `all`: Quote all values. • `minimal`: Optimized minimal quoting of values. • `none`: No quoting of any values. • `nonnumeric`: Quote only non-numeric values. The default is `nonnumeric`.
[`-h` | `--help`]	Displays help on syntax and usage for this command.

Examples

Example A.48 Listing the services and associated deployment groups

Using the shell:

```
(kollacli) service list
```

Using the operating system command prompt:

```
$ kollacli service list
```

Example A.49 Listing the services and associated deployment groups in YAML format

Using the shell:

```
(kollacli) service list -f yaml
```

Using the operating system command prompt:

```
$ kollacli service list -f yaml
```

Example A.50 Listing the services in CSV format with formatting options

Using the shell:

```
(kollacli) service list -f csv -c Service --quote minimal
```

Using the operating system command prompt:

```
$ kollacli service list -f csv -c Service --quote minimal
```

Example A.51 Listing the services and associated deployment groups in table format with formatting options

Using the shell:

```
(kollacli) service list --max-width 60
```

Using the operating system command prompt:

```
$ kollacli service list --max-width 60
```

A.1.1.27 service listgroups

Lists all the services and shows the deployment groups with which they are associated.

Syntax

`service listgroups` [{ `-f` | `--format` } *`file_options`*] [{ `-c` | `--column` } *`column_name`* ...] [`--max-width` *`max_width`*] [`--quote` *`quote_options`*] [`-h` | `--help`]

Where *`file_options`* is:

{ `csv` | `html` | `json` | `table` | `value` | `yaml` }

And *`quote_options`* is:

{ `all` | `minimal` | `none` | `nonnumeric` }

Description

This command lists all the services and shows the deployment groups with which they are associated. This command also shows services which are not part of any group.

Options

The following table shows the available options for this command.

Option	Description
{ `-f` | `--format` } *`file_options`*	Allows you to format the output in various formats. The *`file_options`* option may be one of: • `csv`: Comma Separated Values. • `html`: HTML `table` markup.

Option	Description
	• `json`: JavaScript Object Notation. • `table`: Simple ASCII display table. • `value`: Space separated values with no headers. This format may be useful to pipe output to an operating system command. • `yaml`: YAML format is used natively by Ansible playbooks. The default format is `table`.
{ `-c` | `--column` } *`column_name`* ...	Allows you to specify which column(s) to lists. The *`column_name`* option may be either `Service`, or `Groups` (column names are case sensitive). You can repeat this option to include multiple columns. The default is to include all columns.
`--max-width` *`max_width`*	Sets the maximum display width for each column when using `table` output (`table` is the default output). The *`max_width`* must be an integer. The default for *`max_width`* is `0` (no maximum width).
`--quote` *`quote_options`*	Sets the quoting style for when using *`csv`* output. The *`quote_options`* option may be one of: • `all`: Quote all values. • `minimal`: Optimized minimal quoting of values. • `none`: No quoting of any values. • `nonnumeric`: Quote only non-numeric values. The default is `nonnumeric`.
[`-h` | `--help`]	Displays help on syntax and usage for this command.

Examples

Example A.52 Listing the services and associated deployment groups

Using the shell:

```
(kollacli) service listgroups
```

Using the operating system command prompt:

```
$ kollacli service listgroups
```

Example A.53 Listing the services and associated deployment groups in YAML format

Using the shell:

```
(kollacli) service listgroups -f yaml
```

Using the operating system command prompt:

```
$ kollacli service listgroups -f yaml
```

Example A.54 Listing the services in CSV format with formatting options

Using the shell:

```
(kollacli) service listgroups -f csv -c Service --quote minimal
```

Using the operating system command prompt:

```
$ kollacli service listgroups -f csv -c Service --quote minimal
```

Example A.55 Listing the services and associated deployment groups in table format with formatting options

Using the shell:

```
(kollacli) service listgroups --max-width 20
```

Using the operating system command prompt:

```
$ kollacli service listgroups --max-width 20
```

A.1.1.28 service removegroup

Removes a deployment group from an OpenStack service.

Syntax

`service removegroup` *`service_name`* *`group_name`* [`-h` | `--help`]

Description

This command removes a deployment group from an OpenStack service.

Options

The following table shows the available options for this command.

Option	Description
`service_name`	The OpenStack service name. To show a list of available OpenStack services, and the groups to which they belong, use the service list command.
`group_name`	The name of the deployment group.
[`-h` | `--help`]	Displays help on syntax and usage for this command.

Examples

Example A.56 Removing a service from a deployment group

Using the shell:

```
(kollacli) service removegroup neutron mygroup
```

Using the operating system command prompt:

```
$ kollacli service removegroup neutron mygroup
```

A.1.1.29 setdeploy

Sets whether to deploy Oracle OpenStack for Oracle Linux on local host or on remote hosts.

Syntax

`setdeploy { local | remote } [ -h | --help ]`

Description

This command sets whether to install Oracle OpenStack for Oracle Linux on the local host only, or on remote hosts.

Options

The following table shows the available options for this command.

Option	Description
`{ local | remote }`	Sets whether to deploy Oracle OpenStack for Oracle Linux on local host or on other nodes. Setting this to `local` may be useful during testing to deploy on the local host only. When you want to perform the full deployment on other hosts, set this to `remote`. If this is set to `local`, there can only be one host defined. The default mode is `remote`.
`[ -h | --help ]`	Displays help on syntax and usage for this command.

Examples

Example A.57 Set to deploy Oracle OpenStack for Oracle Linux on local host only

Using the shell:

```
(kollacli) setdeploy local
```

Using the operating system command prompt:

```
$ kollacli setdeploy local
```