FUJITSU

FUJITSU Software ServerView Cloud Load Control V1.0

Installation Guide

J2UL-2088-01ENZ0(00)
November 2015

Contents

About this Manual

This manual describes how to install FUJITSU Software ServerView Cloud Load Control - hereafter referred to as Cloud Load Control (CLC).

This manual is structured as follows:

Chapter	Description
Introduction on page 6	Introduces CLC and its components.
Prerequisites and Preparations on page 8	Describes the prerequisites that must be fulfilled and the preparations you need to take before installing CLC.
Automatically Installing CLC on page 10	Describes how to use the CLC installer.
Manually Installing CLC on page 14	Describes how to manually install the individual CLC components.
Automatically Uninstalling CLC on page 26	Describes how to automatically uninstall the CLC components.
Manually Uninstalling CLC on page 28	Describes how to manually uninstall the CLC components.
Updating CLC Components on page 30	Describes how to proceed when updated CLC components are available.
Open Source Software on page 31	Provides a list of the Open Source software included in and used by CLC.
Glossary on page 32	Definitions of terms used in this manual.

Intended Audience

This manual is directed to people who want to install CLC in an OpenStack environment (OpenStack operators). It assumes that you are familiar with the CLC concepts as described in the *Overview* manual, as well as with the following:

- OpenStack administration and operation
- Administration of Linux-based systems

Notational Conventions

This manual uses the following notational conventions:

Add	The names of graphical user interface elements like menu options are shown in boldface.
`init`	System names, for example command names and text that is entered from the keyboard, are shown in Courier font.
`<variable>`	Variables for which values must be entered are enclosed in angle brackets.

`[option]`	Optional items, for example optional command parameters, are enclosed in square brackets.
`one | two`	Alternative entries are separated by a vertical bar.
`{one | two}`	Mandatory entries with alternatives are enclosed in curly brackets.

Abbreviations

This manual uses the following abbreviations:

CentOS	Community ENTerprise Operating System
CLC	Cloud Load Control
HOT	Heat orchestration template
IaaS	Infrastructure as a Service
KVM	Kernel-based Virtual Machine
OSS	Open Source software
PaaS	Platform as a Service
SaaS	Software as a Service

Available Documentation

The following documentation on CLC is available:

- *Overview* - A manual introducing CLC. It is written for everybody interested in CLC.
- *Installation Guide* - A manual written for OpenStack operators who install CLC in an existing OpenStack environment.
- *Cluster Management Guide* - A manual written for cluster operators who use CLC for provisioning and managing clusters, and for cluster users who deploy applications to a cluster.

Related Web References

The following Web references provide information on open source offerings integrated with CLC:

- *OpenStack*: Documentation of OpenStack, the underlying platform technology.
- *OpenStack Horizon*: Documentation of the OpenStack Horizon dashboard.
- *Kubernetes*: Information on Kubernetes, the core of CLC.
- *Docker*: Information on the container technology used by Kubernetes.

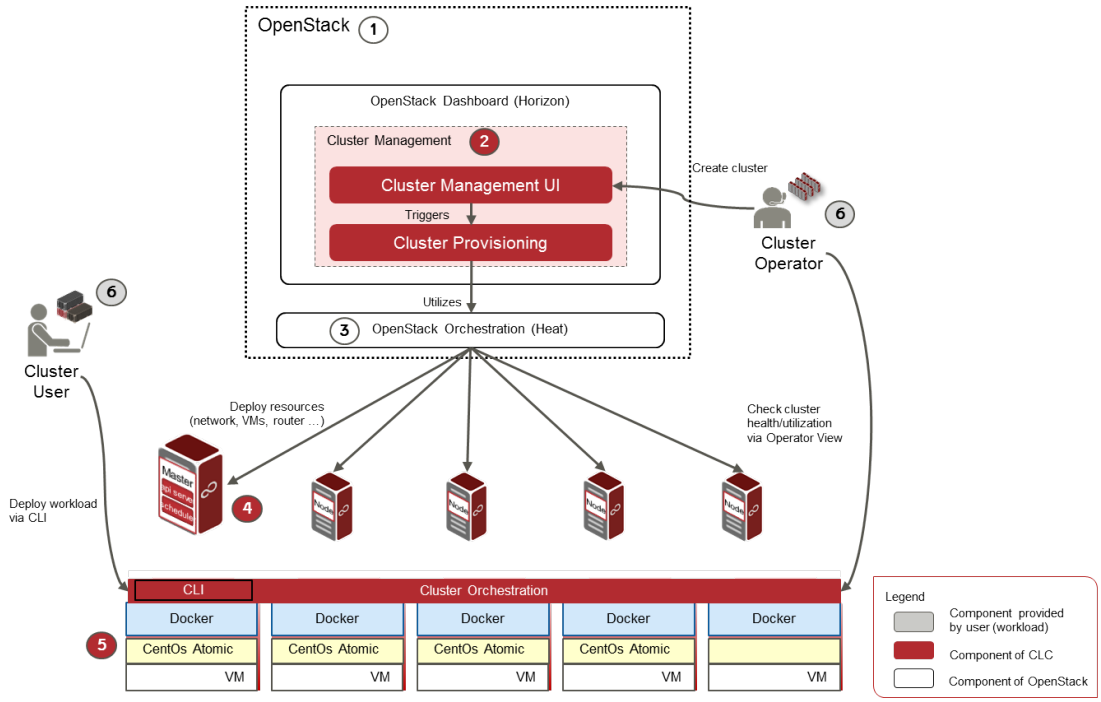
More detailed Web references provided in this manual are subject to change without notice.

1 Introduction

Cloud Load Control (CLC) is an out-of-the-box solution for companies that are looking for a preassembled and enterprise-ready distribution of tools for provisioning and managing clusters and containers on top of OpenStack-based cloud computing platforms. CLC automates the setup and operation of a workload management system in OpenStack based on modern Linux container and clustering technology: *Kubernetes* and *Docker*.

1.1 CLC Components and Their Interaction

The following illustration provides an overview of the CLC components as well as their integration with the underlying platform.



1. **OpenStack** is the underlying platform technology of CLC. It allows for the deployment and management of Infrastructure as a Service (IaaS) platforms.

   For details, refer to the *OpenStack* documentation.

2. After installation of CLC, the **Cluster Management** component is set up, which provides all artefacts and tools required for provisioning a cluster. Cluster Management utilizes the resources provided by the underlying OpenStack platform.

   It comprises a **Cluster Management UI** and **Cluster Provisioning**.

   The Cluster Management UI is a graphical user interface integrated in the OpenStack dashboard for the cluster operator to request the automatic setup of a cluster and to manage clusters on an infrastructure level. It is implemented as a plugin that extends the OpenStack dashboard (**Horizon plugin**).

   Cluster Provisioning allows for the automatic set up of clusters based on resources provided by OpenStack and to set up Cluster Orchestration (Kubernetes). Based on the input values

specified through the Cluster Management UI, instructions for provisioning the cluster are created and passed to the OpenStack Orchestration (**Heat**) system.

3. Predefined Heat orchestration templates (HOT) are part of the CLC Horizon plugin. Heat is responsible for the actual creation of the resources (for example, VMs, volumes, etc.) as well as the installation and configuration of the Cluster Orchestration on top of the created resources.
4. A cluster always comprises at least one master and one worker node. The master is the controlling unit in the cluster and serves as the main management contact point for administrators. Worker nodes communicate with the master, configure the networking for containers, and run the actual workloads assigned to them. The **Docker service** is used to run encapsulated application containers in a relatively isolated but lightweight operating environment.
5. As part of the CLC installation, a preassembled **CentOS Atomic** host image is uploaded and registered in the OpenStack environment. Each node in the cluster is provisioned with this base image.
6. The command-line interface (CLI) of the Cluster Orchestration is used for maintenance and troubleshooting purposes, as well as for deploying the actual workloads onto the cluster nodes. The CLC installation package contains the latest binaries of the CLI so that interaction with the master is instantly possible.

As a summary, CLC comes with

- A preassembled CentOS Atomic host base image for the cluster nodes
- A plugin for the OpenStack dashboard (Horizon plugin) including Heat orchestration templates for provisioning clusters
- Binaries for using the Kubernetes command-line interface

1.2 Installation Options

The following installation scenarios are available:

- **Productive installation** to an OpenStack environment. If you want to install CLC in an environment with one Horizon host only, you can use the automated setup for CLC as described in *Automatically Installing CLC* on page 10. In other OpenStack environments, or if you do not want to run the automatic installation script on your OpenStack instance, you need to follow the manual step-by-step instructions provided in *Manually Installing CLC* on page 14.
- **Evaluation installation**: For a stand-alone evaluation of CLC, a setup based on Vagrant is provided. With this setup, a DevStack-based OpenStack environment is created, and CLC is automatically installed on top of it. If you would like to try this installation, please contact your Fujitsu Support organization.

2 Prerequisites and Preparations

The prerequisites described in this section must fulfilled before automatically or manually installing the CLC Horizon plugin with the Heat orchestration templates, and the preassembled CentOS base image for the cluster nodes.

In addition, preparing the CLC tools and artefacts for installation is required.

2.1 Prerequisites

Access Rights and Privileges

You must have:

- Access to a fully functional OpenStack platform. Access as a user with administrator privileges is required.
- Access to the host on which the OpenStack Horizon service is installed. Access as a user with superuser privileges is required.
- Internet access without proxy settings, in particular access to *Google Container Registry* and *Docker Hub*. This is required because when an application is provisioned in a Kubernetes cluster, Kubernetes tries to download a Docker container image.

System Requirements

CLC can be installed to the following OpenStack platform (OSP):

- Red Hat Enterprise Linux OSP 7

  For a list of certified operating systems working with Red Hat Enterprise Linux OSP 7, refer to the official Red Hat Web site.
- OSP 7 must be configured to use hardware-based virtualization such as KVM (kernel-based virtual machine). Make sure that no emulation hypervisor such as QEMU is configured.
- The OSP 7 orchestration module (Heat) must be installed on the machine you want to use for provisioning clusters.
- CLC supports the following Web browsers:
  - Google Chrome 45.0
  - Microsoft Internet Explorer 11.0
  - Mozilla Firefox 40.0

Physical Resources

In OpenStack, flavors are used for defining the compute, memory, and storage capacity of a virtual server. As such, a flavor is a virtual hardware template. When a cluster is provisioned, CLC creates the nodes of the cluster. The nodes are instances in OpenStack.

The minimum configuration for the nodes provisioned with the preassembled CentOS Atomic base image is as follows:

- 1 vCPU must be present for each node in the cluster. In your OpenStack environment, this vCPU must not be shared with other VMs.
- 1024 MB of RAM virtual machine memory.
- 10 GB of virtual root disk size. This is an ephemeral disk the base image is copied into. You do not use it when you boot from a persistent volume.

2.2 Preparing the CLC Tools and Artefacts

The CLC tools and artefacts are provided on the CLC installation medium, and come as a folder structure on the file system.

Copy this folder structure to a separate folder on the host where you want to install CLC.

In case of an automatic installation, this can be any host from which SSH access is possible to the host where the OpenStack dashboard (Horizon) is installed. The CLC installer copies all required files to the Horizon host. In case of a manual installation, this must be the host where Horizon is installed.

In the following sections, this folder is referred to as `<InstDir>`. It consists of the following subfolders:

- `auto-installation` folder:

  Contains all artefacts required for automatically installing CLC to an existing OpenStack environment.

  For details, refer to *Automatically Installing CLC* on page 10.

- `manual-installation` folder:

  Contains all artefacts required for manually installing CLC step-by-step to an existing OpenStack environment.

  `/horizon-wlm-plugin`: CLC plugin for the OpenStack Horizon dashboard including Heat orchestration templates (HOT) for the automatic orchestration of Kubernetes clusters.

  `/image`: Atomic CentOS base image for the cluster nodes.

  For details, refer to *Manually Installing CLC* on page 14.

- `evaluation-installation` folder:

  Contains all artefacts required for the stand-alone evaluation installation of CLC by means of a setup based on Vagrant. With this setup, a DevStack-based OpenStack environment is created, and CLC is automatically installed on top of it. If you would like to try this installation, contact your Fujitsu Support organization.

- `examples` folder:

  Contains sample files for a guestbook tutorial which walks you through building a simple, multi-tier Web application and deploying it to a Kubernetes cluster.

  For a detailed description of the sample, refer to the *Cluster Management Guide*.

- `doc` folder:

  Contains the user documentation for CLC.

- `cli` folder:

  Contains the binaries of the Kubernetes command-line interface (CLI) which you can use for working with the nodes in a Kubernetes cluster after having installed CLC.

- `licenses` folder:

  Third-party and open source software licenses CLC uses and is based on.

3 Automatically Installing CLC

The automatic installation of CLC consists of the following steps:

1. Preparing the installation environment.
2. Executing the installation script.

Before starting with these steps, make sure that the prerequisites and preparations as described in *Prerequisites and Preparations* on page 8 are fulfilled.

3.1 Preparing the Installation Environment

Before you run the script for automatically installing CLC in your OpenStack environment, prepare the installation environment on the host where the OpenStack Horizon service is installed as described in the following sections.

Note:	You can only use the script if you are operating an OpenStack environment with one Horizon host only.

Ansible 1.9

Ansible is required for installing the CLC software components. Ansible is a simple IT automation platform that makes applications and systems easier to deploy. With Ansible, you avoid writing scripts or custom code to deploy and update applications. Ansible is part of your Red Hat Enterprise Linux environment.

The automated installation of CLC is based on an Ansible playbook, and has been tested using Ansible Version 1.9.4.1.

To check the version:

```
ansible --version
```

Refer to the Red Hat Enterprise Linux Web site for available updates, if required.

Preparing Configuration Settings

Before you can run the CLC installer, you need to specify several configuration settings for identifying your OpenStack environment in the `wlm-config.yaml` configuration file.

You need the following information at hand:

- **IP address** of your environment's **DNS server**. By default, Google's 8.8.8.8 IP address is used.
- Credentials for the **Glance client** configuration:

  The CLC installation procedure registers the CentOS Atomic host image using the OpenStack Glance client. You must have the appropriate user credentials so that the OpenStack Glance client can make queries against your OpenStack cloud. You need to set the required environment variables for the OpenStack Glance client.

  To check the user credentials in OpenStack:

  1. Login to the OpenStack dashboard as an administrator.
  2. Select the project for which you want to install and register the CLC CentOS Atomic host image and which you want to use for provisioning clusters.
  3. Go to **Project > Compute > Access & Security > API Access** and click **View Credentials**.

4. Note down the following information:
   - **Authentication URL** (`os_auth_url`): The URL of your Keystone authentication server.
   - **Project ID** (`os_tenant_id`): The ID of the selected OpenStack project.
   - **User Name** (`os_username`): The name of the user who is to install CLC.

   In addition, you need the **OpenStack password** of the installing user.

Now you can proceed with editing the `wlm-config.yaml` file and specifying the parameters matching your environment:

1. Go to the `<InstDir>/auto-installation` folder.
2. Edit the `wlm-config.yaml` configuration file and specify the following:
   - `horizon_directory`:

     The root folder where Horizon is installed (`<horizon_root_folder>`. The path to the Horizon root folder must be valid and existing in your OpenStack environment.
   - `dns_server_ip`:

     The IP address of the DNS server used by the master and worker nodes. This address is checked by the CLC installation procedure whether domain names can be resolved.
   - `os_auth_url`:

     The URL of your Keystone authentication server. This URL can be looked up in your OpenStack environment (**Project > Compute > Access & Security > API Access**).
   - `os_tenant_id`:

     The ID of the OpenStack project you use for the provisioning of clusters. This ID can be looked up in your OpenStack environment (**Project > Compute > Access & Security > API Access**).
   - `os_username`:

     The OpenStack name of the user who installs CLC. This name can be looked up in your OpenStack environment (**Project > Compute > Access & Security > API Access**).
   - `os_password_input`: The OpenStack password of the user who installs CLC.

Note: The values in the `wlm-config.yaml` file must follow the YAML syntax rules. It is recommended to always enclose user names and passwords with double quotes, since they may contain special characters.

Example:

```
os_username: "myUser5"
os_password_input: "My#secure_password123"
```

In addition, the `wlm-config.yaml` file must not contain empty or unspecified values.

3. Save the configuration file.
4. Make a backup of the configuration file. You need this file when, at a later point in time, you want to uninstall CLC using the CLC uninstaller.

   For details, refer to *Automatically Uninstalling CLC* on page 26.

Example configuration file:

```
#<horizon_root_folder>
horizon_directory: /usr/share/openstack-dashboard

#IP address of DNS server used by master and worker nodes
dns_server_ip: 10.141.19.52

### Glance Authentication Configuration

#Authentication URL
os_auth_url: http://203.0.113.10:5000/v2.0

#Project ID
os_tenant_id: 98333aba48e756fa8f629c83a818ad57

#OpenStack admin user
os_username: "admin"

#OpenStack admin password
os_password_input: "secret"
```

3.2 Executing the Automatic Installation

Once you have prepared your host and the `wlm-config.yaml` configuration file, run the CLC installer as follows:

1. Go to the `<InstDir>/auto-installation` folder.
2. You need the following information at hand:
   - IP address or host name of the OpenStack host where the Horizon service is running.
   - The private key or the password of the installing user to be able to access the OpenStack host via SSH.
3. Run the `wlm-playbook.yaml` Ansible playbook:

```
ansible-playbook \
   -i <horizon host>, \
   -u <login user> \
      --private-key <path of openstack ssh-key> | -k \
   [-b --become-user <OpenStack CLI User> \
       --become-method=sudo --ask-become-pass \]
   wlm-playbook.yaml
```

The options above have the following meaning:

- `-i <horizon host>`
  - Specify the IP address or host name of your the OpenStack host on which the Horizon service is running.
  - Make sure to enter a comma after the IP address or host name, and to enter a blank after the comma.
- `-u <login user> --private-key <path of openstack ssh-key> | -k`
  - Specify the user name of the OpenStack administrator running the installation script.

  - The path of the OpenStack SSH key is the path to the file containing the private key of the above administrator and is required so that the administrator can access the OpenStack instance via SSH.
  - If you want to use a password, specify the `-k` option instead of `--private-key`. You will be prompted for the SSH password of the `login user` instead of assuming key-based authentication with the SSH agent.
- `[-b --become-user <OpenStack CLI User> --become-method=sudo --ask-become-pass]`
  - Optionally, specify a user who will later be able to execute commands from the OpenStack command-line interface (CLI). This user will have superuser privileges (for example, the root user) in OpenStack.
  - If the system is configured such that `sudo` commands require a password, the `--ask-become-pass` option ensures that the root user will be prompted for his password.
  - If you omit the options, the user specified with the `-u` option will be able to execute OpenStack CLI commands.

Example 1:

```
ansible-playbook \
   -i 10.234.10.300, \
   -u admin --private-key usr/bin/admin.key \
   wlm-playbook.yaml
```

Example 2:

```
ansible-playbook \
   -i 10.140.18.49, \
   -u osp7 \
   -k \
   -b --become-user root --become-method=sudo --ask-become-pass \
   wlm-playbook.yaml
```

Results

The automatic installation of CLC has the following results:

- The CLC Horizon plugin is installed to the `/wlm` subfolder of the Horizon root folder. It adds an additional panel to the OpenStack dashboard. The plugin installation also adds CLC Heat orchestration templates, which are later used to provision a Kubernetes cluster.
- The preassembled CLC CentOS Atomic image is uploaded and registered in the OpenStack environment. This image is the base image for cluster nodes: `WLM:Atomic`.

Note: When the installation process is complete, check the console output of the shell from which you executed the Ansible playbook. For example, the CLC installer checks whether the specified IP address of the DNS server is valid and whether the DNS server can be reached. If this is not the case, warnings are output to the console. Check and possibly correct the `wlm-config.yaml` configuration file, and run the Ansible playbook again.

4 Manually Installing CLC

The manual installation of the CLC components consists of the following steps:

1. Registering the preassembled CentOS Atomic host image on your OpenStack host.
2. Installing the CLC plugin for the OpenStack dashboard including the CLC Heat orchestration templates.

Before starting with these steps, make sure that the prerequisites as described in *Prerequisites and Preparations* on page 8 are fulfilled.

4.1 Registering the CentOS Atomic Host Image

As a first step in the manual installation of CLC, you need to upload and register the preassembled CLC Atomic host base image for the cluster nodes in your OpenStack instance.

You can use the OpenStack dashboard or the OpenStanck `glance` command for registering the preassembled base image for cluster nodes.

The image is provided in the `<InstDir>/manual-installation/image` folder.

Registering the Image Using the OpenStack Dashboard

To register the CentOS Atomic host image using the OpenStack dashboard, proceed as follows:

1. Login to the OpenStack dashboard as an administrator.
2. Go to **Admin > System > Images**.
3. Click **Create Image** and create the image with the following parameters:
   - **Name**: `WLM:Atomic` (mandatory). Do not specify a different name!
   - **Description** of the image (optional)
   - **Image Source**: Select **Image File** from the drop-down list and browse for the image file. The file is located in the `<InstDir>/manual-installation/image` folder and called `centos-atomic-host-7.qcow2`.
   - **Format**: `QCOW2 - QEMU Emulator` (mandatory). Do not specify a different disk format!
   - **Minimum Disk (GB)**: 10 (recommended because this value is used to filter out flavors that are not large enough for clusters. A flavor must be chosen when a cluster is created.)
   - **Minimum RAM (MB)**: 1024 (recommended - see above)
   - Select **Public** to make the image public to all users (mandatory).
4. Click **Create Image**.

The image is queued to be uploaded. It may take several minutes before the status changes to `Active`.

Registering the Image using the CLI

You must have the appropriate credentials for the Glance client configuration if you want to use the OpenStack Glance client to make queries against your OpenStack cloud. For registering the CLC CentOS Atomic host image using the Glance client, you need to set the required environment variables for the Glance client.

To do so, you must create an environment file called OpenStack RC file or `openrc.sh` file. This project-specific environment file contains the credentials that all OpenStack services use. When

you source the file, environment variables are set for your current shell. The variables enable the OpenStack client commands to communicate with the OpenStack services that run in the cloud.

To set the environment variables, download, and source the OpenStack RC file:

1. Log in to the OpenStack dashboard as an administrator.
2. Select the project for which you want to download the OpenStack RC file, and click **Project > Compute > Access & Security**.
3. On the **API Access** tab, click **Download OpenStack RC File** to generate the environment file for in your shell with the environment variables the Glance client requires to know where your service endpoints and your authentication information are.
4. Save the file to the OpenStack host where the Horizon service is running, for example, to the `<InstDir>` folder.

   The file name is `<project>-openrc.sh`, where `<project>` is the name of the project for which you downloaded the file.
5. On the shell from which you want to execute OpenStack commands, source the `<project>-openrc.sh` file for the respective project.

   In the following example, the file is sourced for the `admin` project:

   ```
   source admin-openrc.sh
   ```
6. When you are prompted for an OpenStack password, enter the password of the user who downloaded the `<project>-openrc.sh` file.

Below you find a sample file, `admin-openrc.sh`.

```
export OS_AUTH_URL=http://203.0.113.10:5000/v2.0
export OS_TENANT_ID=98333aba48e756fa8f629c83a818ad57
export OS_TENANT_NAME="admin"
export OS_USERNAME=demo
read -s OS_PASSWORD_INPUT
export OS_PASSWORD=$OS_PASSWORD_INPUT
```

where:

- `OS_AUTH_URL` is the URL of your Keystone authentication server.
- `OS_TENANT_ID` is the ID of the project you use for the provisioning of clusters.
- `OS_TENANT_NAME` is the name of the project you use for the provisioning of clusters.
- `OS_USERNAME` is the OpenStack name of the user who installs CLC.
- `read -s OS_PASSWORD_INPUT`

  `export OS_PASSWORD=$OS_PASSWORD_INPUT`

  The password is not stored as plain text. When you source or run the script, it prompts you for your password and stores it in the environment variable `OS_PASSWORD`. It is important to note that this requires interaction. It is possible to store a value directly in the script if you require a non-interactive operation, but you then need to be extremely cautious regarding the security and permissions of this file.

To register the CentOS Atomic host image:

Execute the following `glance` command:

```
glance image-create --name=WLM:Atomic --min-disk=10 --min-ram=1024 \
  --container-format=bare --disk-format=qcow2 \
  --file <Atomic host imagefile>  --is-public true
```

where:

- `name=WLM:Atomic` is mandatory. Do not specify a different name!
- `min_disk` and `min_ram` are recommended because these values are used to filter out flavors that are not large enough for clusters. A flavor must be chosen when a cluster is created. The values define the minimum disk space and RAM allocated for the image.
- `container-format=bare` is the default for Docker containers.
- `disk-format=qcow2` is mandatory. Do not specifiy a different disk format.
- `file` is the CentOS Atomic host image file located in `<InstDir>/manual-installation/image`
- `is-public true` is mandatory.

To confirm that the image was registered:

Execute the following `glance` command:

```
glance image-list
```

This command shows the attributes of the newly created CLC image.

4.2 Installing the CLC Horizon Plugin

Proceed as follows to install the CLC plugin for the OpenStack dashboard:

1. Log in as a superuser to the OpenStack host where the dashboard is installed.
2. Copy the following folder to the folder on your OpenStack host where the dashboard is installed (`<horizon-root-folder>`):

   `<InstDir>/manual-installation/horizon-wlm-plugin/wlm`

   For example:

   ```
   scp -r horizon-wlm-plugin/wlm /usr/share/openstack-dashboard/
   ```

   Note: Do not move the folder (`mv`) but copy it!

3. Set the access rights for this folder to `755`:

   ```
   sudo chmod -R 755 <horizon-root-folder>/wlm
   ```

   In this way, the folder contents can be listed and read by all users, and all files in it can be executed.

4. Copy the `wlm.py` file from the `<InstDir>/manual-installation/` folder to the following folder on your OpenStack host where the dashboard is installed:

   `<horizon-root-folder>/openstack_dashboard/local/enabled/`

   The `wlm.py` file defines the location where to add the CLC-specific panel to the OpenStack dashboard, the location of the Heat orchestration template for the automatic provisioning of clusters, as well as the IP address of the DNS server of your environment.

5. Adapt the following values in the copied `wlm.py` file:

```
WLM_HEAT_TEMPLATE_FILE_PATH=
  <horizon-root-folder>/wlm/heat-provisioner/kubecluster.yaml

DNS_SERVER_IP_ADDRESS=<IP of your DNS server>
```

   The `kubecluster.yaml` Heat template is used to provision a Kubernetes cluster with one or more nodes (default: 2). The cluster uses Flannel to provide an overlay network connecting Kubernetes pods deployed on different nodes. Pods are the smallest deployable units that can be created, scheduled, and managed by Kubernetes.

6. Since the CLC-specific panel added to the OpenStack dashboard requires some Javascript and a style sheet so that it can be rendered completely, you need to collect these static files and add them to your OpenStack environment.

   Execute the following command in the `<horizon-root-folder>` folder:

```
sudo ./manage.py collectstatic
```

7. For finishing the integration of the CLC plugin into the dashboard, restart the Horizon service.

   For example:

```
sudo service httpd restart
```

5 Next Steps

After having successfully installed CLC, continue with the following:

1. Make the binaries for using the Kubernetes command-line interface (CLI) publicly available.
2. Create a custom flavor for the provisioning of clusters.
3. Create a key pair for SSH access.
4. Check the CLC installation in the OpenStack dashboard.

5.1 Preparing the Kubernetes CLI

To deploy and manage applications on Kubernetes, you use the Kubernetes command-line tool, `kubectl`. It lets you inspect your cluster resources, create, delete, and update components, and much more. You will use it to look at your new cluster and bring up example applications.

So, before you can start with the deployment of workloads to the cluster nodes, you need to make the Kubernetes command-line interface (CLI) available. The binaries of the CLI are available in the `<InstDir>/cli` folder.

To be able to execute the `kubectl` commands from any folder, link `kubectl` to your `/usr/local/bin` folder, for example:

```
sudo ln -s <InstDir>/cli/kubectl /usr/local/bin/kubectl
```

To manage your Kubernetes environment, and to deploy workloads into the containers, you run the `kubectl` commands. `kubectl` controls the Kubernetes cluster manager.

For a complete list of the parameters that are supported, you can execute the following command:

```
kubectl --help
```

For details on the parameters for `kubectl`, you can also refer to *Google's Kubernetes CLI description*.

Note: If the `kubectl` command cannot be executed, the executable bit might not be set for the file. You can check this, for example, by using the `ls -l` command. If the executable bit is not set, you can set it with the following command:

```
chmod +x kubectl
```

For samples of how to use the `kubectl` commands, refer to the *Cluster Management Guide*.

5.2 Defining a Custom Flavor

Flavors are virtual hardware templates in OpenStack, defining sizes for RAM, disks, number of cores, and other resources. The default OpenStack installation provides several flavors. It is recommended to create a flavor which defines the minimum configuration for the creation of a Kubernetes cluster with 2 CentOS Atomic host image nodes.

Proceed as follows:

1. Log in to the OpenStack dashboard as an administrator.
2. Go to **Admin > System > Flavors**.
3. Click **Create Flavor**.

4. In the **Create Flavor** window, on the **Flavor Information** tab, enter or select the parameters for the flavor.
   - **Name**: Enter a name for the flavor.
   - **ID**: Leave this field blank. OpenStack generates the ID.
   - **VCPUs**: Enter the number of virtual CPUs to use. The minimum value is `1`.
   - **RAM (MB)**: Enter the amount of RAM to use in megabytes. The minimum recommended value is `1024`.
   - **Root Disk (GB)**: Enter the amount of disk space in gigabytes to use for the root (/) partition. The minimum recommended value is `10`.
   - **Ephemeral Disk (GB)**: Enter the amount of disk space in gigabytes to use for the ephemeral partition. If unspecified, the value is `0` by default. Ephemeral disks provide local disk storage for the life cycle of a VM instance. When a VM is terminated, all data on the ephemeral disk is lost. Ephemeral disks are not included in any snapshots.
   - **Swap Disk (MB)**: Enter the amount of swap space in megabytes to use. If unspecified, the default is `0`.
5. On the **Flavor Access** tab, you can control access to the flavor: You can specfiy which OpenStack projects can use the flavor by moving projects from the **All Projects** column to the **Selected Projects** column.

   Only projects in the **Selected Projects** column can use the flavor. If there are no projects in this column, all projects can use the flavor.
6. Click **Create Flavor**.

The flavor is added to the list from which you can choose when creating a cluster.

The following table lists the OpenStack default flavors and your newly created one:

Flavor	VCPUs	Disk (in GB)	RAM (in MB)
m1.tiny	1	1	512
<YourFlavor>	1	10	1024
m1.small	1	20	2048
m1.medium	2	40	4096
m1.large	4	80	8192
m1.xlarge	8	160	16384

Note: The CentOS Atomic host image also requires RAM and system resources (about 750 MB). You must take this into account when you create nodes. When you create a cluster, for example, with a flavor defining 1024 MB of RAM, you only have about 275 MB left for running applications in the pods on your nodes. Make sure to have enough resources available when setting up a cluster.

5.3 Creating a Key Pair for SSH Access

Key pairs are SSH credentials that are injected into an OpenStack instance when it is launched. Each OpenStack project should have at least one key pair.

On the host which you use to access the OpenStack dashboard, generate a key pair as follows:

```
ssh-keygen -t rsa -f cloud.key
```

This command generates a private key (`cloud.key`) and a public key (`cloud.key.pub`).

Then proceed as follows:

1. Log in to the OpenStack dashboard as an administrator.
2. Go to **Project > Orchestration > Clusters**.
3. Click **Create Cluster**.
4. On the **Access & Security** tab, click the plus (+) sign.
5. In the **Key Pair Name** field, enter a name for the key pair. It is recommended to use letters, numbers, and hyphens only.
6. Copy the contents of the public key file you just generated, and paste it into the **Public Key** field of the **Import Key Pair** page.
7. Proceed with the provisioning of your cluster as described in *Testing the CLC Installation* on page 20.

When an instance is created in OpenStack, it is automatically assigned a fixed IP address in the network to which it is assigned. This IP address is permanently associated with the node (instance) until it is terminated. However, in addition to the fixed IP address, you also assign floating IP addresses to the cluster nodes when you create a cluster. Unlike fixed IP addresses, floating IP addresses are able to have their associations modified at any time, regardless of the state of the nodes involved.

After launching a node, you can log in to it using the private key:

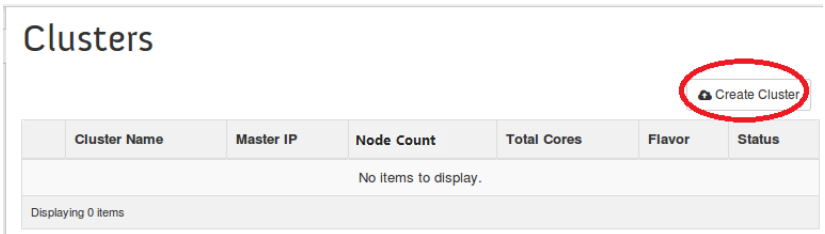
```
ssh -i cloud.key minion@<floating IP of node>
```

5.4 Testing the CLC Installation

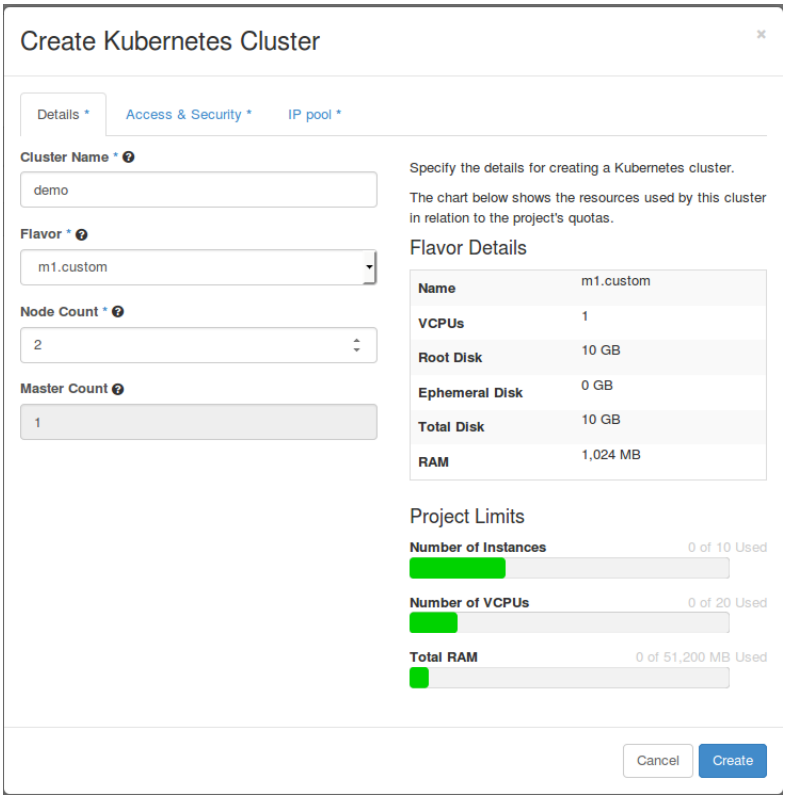
To check whether CLC has been installed correctly, access the **Clusters** panel in the OpenStack dashboard:

1. Log in to the OpenStack dashboard as an administrator.
2. Go to **Project > Orchestration > Clusters**.

   The panel for cluster management provided by CLC is opened:



3. Click **Create Cluster**:



4. On the **Details** tab, enter the following information:
   - **Cluster Name**

     Name of the cluster. The name must be unique within the OpenStack project, start with a letter and only contain alphanumeric characters, numbers, underscores, periods, and hyphens. The cluster name may consist of a maximum of 200 characters.
   - **Flavor**

     Flavors are virtual hardware templates in OpenStack, defining sizes for RAM, disks, number of cores, and other resources. The default OpenStack installation provides several flavors. It is recommended to create and use a flavor which defines the minimum configuration for the creation of a Kubernetes cluster with CentOS Atomic host image nodes:
     - 1 vCPU (virtual CPU) must be present for each node in the cluster. In your OpenStack environment, this vCPU must not be shared with other VMs.
     - 1024 MB of RAM virtual machine memory.
     - 10 GB of virtual root disk size.

     Flavors which are not large enough for cluster environments are filtered out from the list of flavors that can be chosen.

     Refer to *Defining a Custom Flavor* on page 18 for information on how to create a flavor.
   - **Node Count**

Number of worker nodes to be created in the cluster to be provisioned. As an example, specify `2`.

As you type, in the **Project Limits** area, you see the resources the cluster will use in relation to the quota set for the current project.

Quota are operational limits. CLC uses the default OpenStack project quota. Without sensible quota, a single project could exceed all available physical resources, and thus make the whole OpenStack environment unstable. The colors have the following meaning:

- Blue: The resources already used by the number of instances in OpenStack defined in the project's quota before the cluster is created.
- Green: The resources that will be used as soon as the cluster is created.
- Red: The number of nodes you specified will exceed the resources defined in the default quota of the OpenStack project. Thus you either need to increase the project quota, or specify a lower number of nodes in the **Node Count** field.

A Kubernetes cluster always consists of a master node and one or several worker nodes. The master node is added to the amount of nodes that you specify in the **Node Count** field.

- **Master Count** (read-only)

  The number of Kubernetes master nodes. Currently, there can only be 1 master.

5. On the **Key Pair for Access & Security** tab, select a key pair or create a new one by clicking the plus (+) sign.

   Key pairs define how you log in to a cluster node after it has been launched. Cloud images support public key authentication rather than conventional user name/password authentication. Key pairs are SSH credentials that are injected into an OpenStack instance when it is launched. Each project should have at least one key pair.

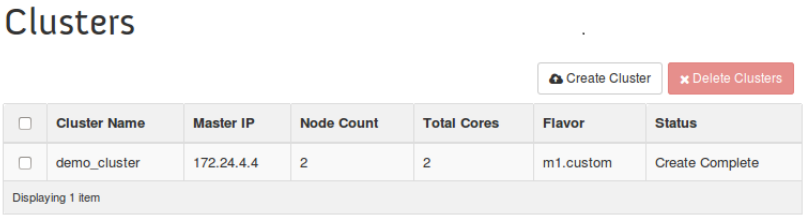
   When using the dashboard for the first time, you need to create a key pair for the project. Refer to *Creating a Key Pair for SSH Access* on page 19 for details on how to create a key pair.

6. On the **IP Pool** tab, select a network.

   An OpenStack administrator can set up several networks and subnets. The network defines the range of floating IP addresses the nodes in the cluster to be provisioned can be assigned.

7. Click **Create**.

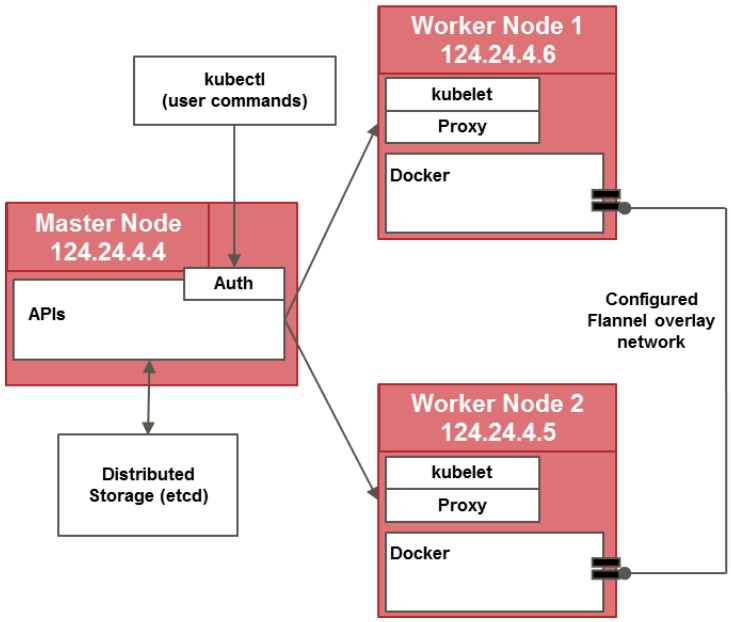
CLC starts provisioning the Kubernetes cluster. The panel for cluster management provided by CLC now looks, for example, as follows:



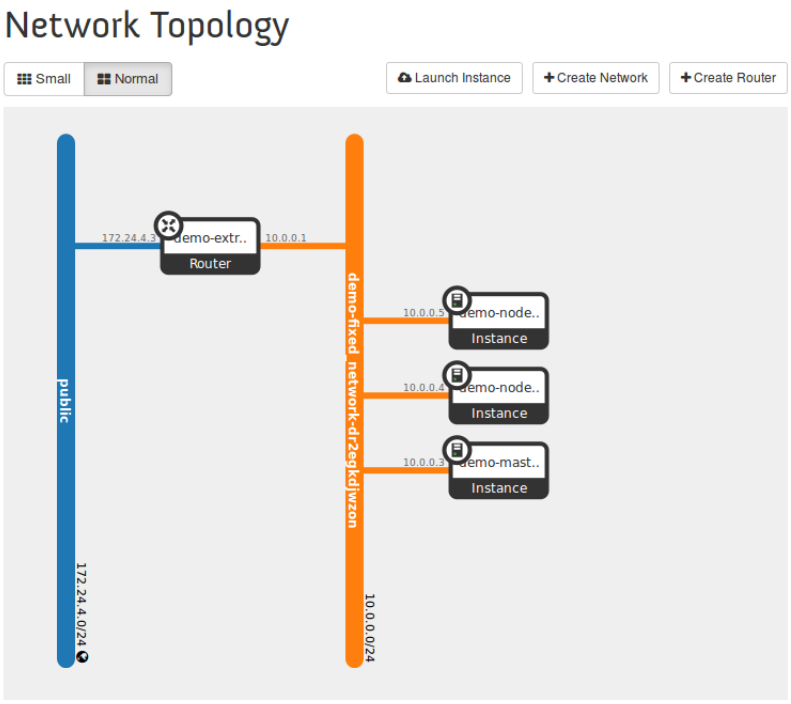
- **Cluster Name**: The name of the cluster as specified when you created the cluster.
- **Master IP**: IP address of the master node.
- **Node Count**: Number of provisioned worker nodes.
- **Total Cores**: Number of vCPU cores used by the provisioned worker nodes.

- **Flavor**: The flavor used for the cluster as selected when you created the cluster.
- **Status**: The status of the cluster. This is, for example, `Create in Progress` when Kubernetes starts to provision the cluster, or `Create Complete` when the provisioning of the cluster is completed.

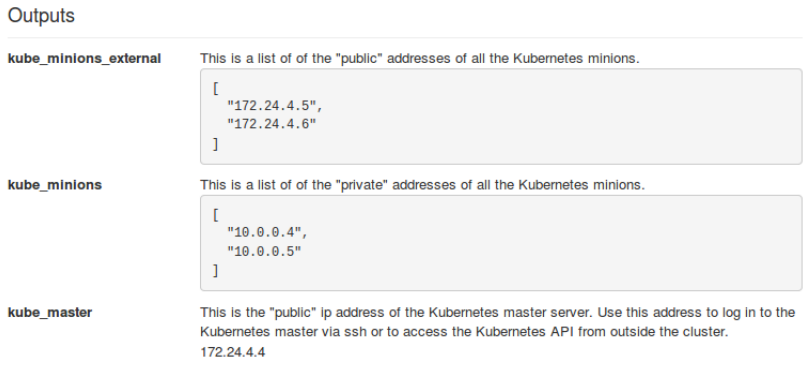
When the provisioning of the cluster is finished, a stack representing the cluster is created in OpenStack. The CLC Heat orchestration template boots the Kubernetes cluster with the specified number of worker nodes and one master node:



The networking between the containers is configured using Flannel, a network fabric layer that provides each node with an individual subnet for container communication. You can see this configuration in the OpenStack dashboard (**Project > Network > Network Topology**):



The newly created stack represents the cluster in OpenStack. When you take a look at the stack (**Project > Orchestration > Stacks**), you see the floating IP addresses of the master and the worker nodes:



The floating IP addresses can also be viewed on an instance level (**Project > Compute > Instances**).

Note:	You can always check the status of the cluster and its health with **Project > Orchestration > Stacks**.

You can now proceed with deploying workloads to the cluster. Refer to the *Cluster Management Guide* for details.

6 Automatically Uninstalling CLC

CLC comes with an Ansible playbook for uninstalling the CLC Horizon plugin, the CentOS Atomic host image, and the Heat orchestration templates.

Note: You need the same configuration file (`wlm-config.yaml`) for uninstalling that you used for automatically installing CLC. For details, refer to *Automatically Installing CLC* on page 10.

Run the CLC uninstaller as follows:

1. Go to the `<InstDir>/auto-installation` folder.
2. You need the following information at hand:
   - IP address or host name of the OpenStack host where the Horizon service is running.
   - The private key or the password of the installing user to be able to access the OpenStack host via SSH.
   - The `wlm-config.yaml` configuration file used for automatically installing CLC.
3. Run the `wlm-plugin-uninstall.yaml` Ansible playbook:

```
ansible-playbook \
   -i <horizon host>, \
   -u <login user> \
      --private-key <path of openstack ssh-key> | -k \
   [-b --become-user <OpenStack CLI User> \
       --become-method=sudo --ask-become-pass \]
   wlm-plugin-uninstall.yaml
```

The options above have the following meaning:

- `-i <horizon host>`
  - Specify the IP address or host name of your the OpenStack host on which the Horizon service is running.
  - Make sure to enter a comma after the IP address or host name, and to enter a blank after the comma.
- `-u <login user> --private-key <path of openstack ssh-key> | -k`
  - Specify the user name of the OpenStack administrator running the installation script.
  - The path of the OpenStack SSH key is the path to the file containing the private key of the above administrator and is required so that the administrator can access the OpenStack instance via SSH.
  - If you want to use a password, specify the `-k` option instead of `--private-key`. You will be prompted for the SSH password instead of assuming key-based authentication with the SSH agent.
- `[-b --become-user <OpenStack CLI User> --become-method=sudo --ask-become-pass]`
  - Specify the user who is able to execute commands from the OpenStack command-line interface (CLI). This user has superuser privileges (the root user) in OpenStack.
  - If the system is configured such that `sudo` commands require a password, the `--ask-become-pass` option ensures that the root user is prompted for his password.

- If you omit the options, the user specified with the `-u` option can execute OpenStack CLI commands.

Example 1:

```
ansible-playbook \
   -i 10.234.10.300, \
   -u admin --private-key usr/bin/admin.key \
   wlm-plugin-uninstall.yaml
```

Example 2:

```
ansible-playbook \
   -i 10.140.18.49, \
   -u osp7 \
   -k \
   -b --become-user root --become-method=sudo --ask-become-pass \
   wlm-plugin-uninstall.yaml
```

Results

- The CLC Horizon plugin and the CLC Heat orchestration templates are removed.
- The CentOS Atomic host image for cluster nodes is deregistered and removed from your OpenStack environment.
- The `/wlm` subfolder to which the CLC Horizon plugin was installed is removed.
- Existing clusters are not affected. The nodes can still be accessed using `kubectl` commands.

Note: If the `wlm-config.yaml` configuration file contains incorrect values, for example, a non-existing path to where the OpenStack dashboard is installed, the installation procedure is aborted. You can then correct the `wlm-config.yaml` and run the uninstaller again; or you can proceed with manually uninstalling the CLC components as described in *Manually Uninstalling CLC* on page 28.

7 Manually Uninstalling CLC

If you want to manually uninstall CLC, you need to execute the following steps:

1. Deleting the preassembled CentOS Atomic host image on your OpenStack host.
2. Uninstalling the CLC plugin for the OpenStack dashboard including the CLC Heat orchestration templates.

7.1 Deleting the CentOS Atomic Host Image

As a first step in the manual uninstallation of CLC, you need to delete the preassembled CLC Atomic host base image for the cluster nodes in your OpenStack instance.

You can use the OpenStack dashboard or the OpenStanck `glance` command for deleting the preassembled base image for cluster nodes.

Deleting the Image Using the OpenStack Dashboard

To delete the CentOS Atomic host image using the OpenStack dashboard, proceed as follows:

1. Login to the OpenStack dashboard as an administrator.
2. Go to **Admin > System > Images**.
3. Select the `WLM:Atomic` image, and click **Delete Images**.
4. Confirm the deletion of the image.

Deleting the Image using the CLI

You must have the appropriate credentials for the Glance client configuration if you want to use the OpenStack Glance client to make queries against your OpenStack cloud. For deleting the CLC CentOS Atomic host image using the Glance client, you need to set the required environment variables for the Glance client. If you manually installed CLC as described in *Manually Installing CLC* on page 14, you have already created the required environment file, `openrc.sh`. Reuse this file containing the credentials that all OpenStack services use, and for setting the environment variables for your current shell.

To delete the CentOS Atomic host image:

Execute the following `glance` command:

```
glance image-delete WLM:Atomic
```

To confirm that the image was deleted:

Execute the following `glance` command:

```
glance image-list
```

This command no longer shows the CLC image.

7.2 Uninstalling the CLC Horizon Plugin

Proceed as follows to uninstall the CLC plugin from the OpenStack dashboard:

1. Log in as a superuser to the OpenStack host where the dashboard is installed.

2. Delete the folder on your OpenStack host where the CLC Horizon plugin is installed (`<horizon-root-folder>/wlm`).

   For example:

   ```
   rm -rf /usr/share/openstack-dashboard/wlm
   ```

3. Remove the CLC Horizon plugin registry entries `wlm.py` and `wlm.pyc` (compiled version).

   For example:

   ```
   rm -f \
   /usr/share/openstack-dashboard/openstack_dashboard/local/enabled/wlm.py
   ```

   and

   ```
   rm -f \
   /usr/share/openstack-dashboard/openstack_dashboard/local/enabled/wlm.pyc
   ```

4. Since the CLC-specific panel added to the OpenStack dashboard required some Javascript and a style sheet so that it could be rendered completely, you need to clean up these static files and remove them from your OpenStack environment.

   Execute the following command in the `<horizon-root-folder>` folder:

   ```
   sudo ./manage.py collectstatic --noinput
   ```

5. For finishing the deletion of the CLC plugin from the dashboard, restart the Horizon service.

   For example:

   ```
   sudo service httpd restart
   ```

Note: Uninstalling the Horizon plugin does not affect existing clusters. The nodes can still be accessed using `kubectl` commands.

8 Updating CLC Components

Since CLC uses and is based on Open Source software, you might need to update the components of CLC from time to time.

In this case, proceed as follows:

1. If you have data that are stored persistently in your cluster, backup the data.
2. Uninstall CLC as described in *Automatically Uninstalling CLC* on page 26 or *Manually Uninstalling CLC* on page 28.
3. Reinstall CLC either automatically as described in *Automatically Installing CLC* on page 10 or manually as described in *Manually Installing CLC* on page 14.

Existing clusters are not affected unless a new Atomic host image is available. In this case, a new version of CLC is available and you will find descriptions on how to proceed in the documentation of this new version.

Appendix A: Open Source Software

The following Open Source Software (OSS) is included in and used by CLC:

Software	Description	License	
Kubernetes	Container Cluster Manager from Google.	Apache 2.0	*Info*
Docker	Open-source application container engine, used by Kubernetes.	Apache 2.0	*Info*
etcd	A highly available key-value store for shared configuration and service discovery, required by Kubernetes.	Apache 2.0	*Info*
flannel	etcd backend network fabric for containers, required by Kubernetes	Apache 2.0	*Info*
OpenStack Horizon	OpenStack dashboard. CLC provides a plugin for Horizon.	Apache 2.0	*Info*
OpenStack Heat	OpenStack Orchestration. Used by CLC to provision a cluster.	Apache 2.0	*Info*
OpenStack Glance	OpenStack service for discovering, registering, and retrieving virtual machine images. CLC registers the image of the nodes of a cluster with this service.	Apache 2.0	*Info*

The following OSS is used by CLC, but not included in the product:

Software	Description	License	
Ansible	IT automation platform. Ansible scripts are used for the installation of CLC. Ansible must be installed on the host system.	GNU GPL v3	*Info*

Glossary

Cloud

A metaphor for the Internet and an abstraction of the underlying infrastructure it conceals.

Container

A Linux-based technology, where a single kernel runs multiple instances on a single operating system. Containers form an additional layer of abstraction and automation of operating-system-level virtualization.

Cluster

A group of servers and other resources that act like a single system and enable high availability, load balancing and parallel processing.

DevStack

The OpenStack development environment. It can be used to demonstrate starting and running OpenStack services and provide examples of using them from a command line. DevStack has evolved to support a large number of configuration options and alternative platforms and support services.

Docker

An Open Source project that automates the deployment of applications inside software containers. Docker uses resource isolation features of the Linux kernel to allow independent containers to run within a single Linux instance, avoiding the overhead of starting and maintaining virtual machines.

Flavor

A virtual hardware template used in OpenStack for defining the compute, memory, and storage capacity of a virtual server.

Heat

A service to orchestrate multiple composite cloud applications.

Infrastructure as a Service (IaaS)

The delivery of computer infrastructure (typically a platform virtualization environment) as a service.

Label

In the context of cluster orchestration, a facility for organizing and selecting groups of objects based on key-value pairs.

Platform as a Service (PaaS)

The delivery of a computing platform and solution stack as a service.

Pod

An application-specific logical host in a containerized environment. Pods may contain one or more tightly coupled containers.

Pods are the smallest deployable units that can be created, scheduled, and managed by Kubernetes. Pods can be created individually, the usage of *Replication Controllers*, however, is recommended.

Replication Controller

An abstraction which ensures that a specific number of *Pod* replicas are running at all times. If a pod or host goes down, the replication controller ensures that enough pods get recreated elsewhere. A replication controller manages the lifecycle of pods.

Service

In the context of cluster orchestration, a layer of abstraction for providing a single, stable name and address for accessing a set of *Pods*. Services act as basic load balancers.

Software as a Service (SaaS)

A model of software deployment where a provider licenses an application to customers for use as a service on demand.

Stack

A set of OpenStack resources created and managed by the Orchestration service according to a given template (in CLC, a Heat Orchestration Template (HOT)).

Tenant

A group of users in OpenStack. Also referred to as "project".

Workload

Abstraction of the actual amount of work that applications or services perform in a given period of time, or the average amount of work handled by an application or service at a particular point in time.