FUJITSU

FUJITSU Software ServerView Cloud Load Control V1.1

Overview

J2UL-2087-01ENZ0(01)
October 2016

Contents

About this Manual 4
1 What is Cloud Load Control? 7
1.1 Key Features 7
1.2 Components 8
1.3 Users and Roles 9
1.4 Interaction of Components and Users 10
2 Cluster Management 12
3 Cluster Orchestration 13
Appendix A Open Source Software 14
Glossary 15

About this Manual

This manual is an introduction to FUJITSU Software ServerView Cloud Load Control - hereafter referred to as Cloud Load Control (CLC).

This manual is structured as follows:

Chapter	Description
What is Cloud Load Control? on page 7	Introduces CLC, its key features, components, and users.
Cluster Management on page 12	Describes the features and basic concepts of cluster provisioning and management.
Cluster Orchestration on page 13	Describes the features and basic concepts of cluster orchestration.
Open Source Software on page 14	Provides a list of Open Source software used with CLC.
Glossary on page 15	Glossary

Intended Audience

This manual is written for people who have a basic knowledge of OpenStack environments and want to get to know the key features and functions CLC offers on top of OpenStack.

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.

Export Restrictions

Exportation/release of this document may require necessary procedures in accordance with the regulations of your resident country and/or US export control laws.

Trademarks

Trademarks of other companies are used in this manual only to identify particular products or systems.

LINUX is a registered trademark of Linus Torvalds.

UNIX is a registered trademark of the Open Group in the United States and in other countries.

The OpenStack® Word Mark and OpenStack Logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation in the United States and other countries and are used with the OpenStack Foundation's permission.

Red Hat is a trademark or a registered trademark of Red Hat Inc. in the United States and other countries.

Kubernetes is a registered trademark of Google Inc.

Docker is a trademark of Docker Inc.

ServerView is a registered trademark of FUJITSU LIMITED.

Other company names and product names are trademarks or registered trademarks of their respective owners.

Other company and product names in this manual are trademarks or registered trademarks of their respective owners.

Copyrights

High Risk Activity

The Customer acknowledges and agrees that the Product is designed, developed and manufactured as contemplated for general use, including without limitation, general office use, personal use, household use, and ordinary industrial use, but is not designed, developed and manufactured as contemplated for use accompanying fatal risks or dangers that, unless extremely high safety is secured, could lead directly to death, personal injury, severe physical damage or other loss (hereinafter "High Safety Required Use"), including without limitation, nuclear reaction control in nuclear facility, aircraft flight control, air traffic control, mass transport control, medical life support system, missile launch control in weapon system. The Customer shall not use the Product without securing the sufficient safety required for the High Safety Required Use. In addition, FUJITSU (or other affiliate's name) shall not be liable against the Customer and/or any third party for any claims or damages arising in connection with the High Safety Required Use of the Product.

1 What is Cloud Load Control?

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*.

CLC contributes to the Kubernetes open source project initiated by Google. It helps improve this powerful system for managing containerized applications in a cluster environment.

1.1 Key Features

The key features of CLC are:

- Integration of container technology and cluster orchestration
- Automated provisioning and setup of clusters on top of OpenStack
- Seamless integration in OpenStack dashboard

Integration of Container Technology and Cluster Orchestration

Container technology provides for the decoupling of applications from infrastructure, as well as the independence in where an application is deployed. This separation leads to increased efficiency and reliability, in particular to faster and more frequent application deployments. Workloads become mobile and can be shifted quickly between execution environments. This helps to maintain agility and control cost.

Containers are a layer of abstraction and automation on top of operating-system-level virtualization. OS-level virtualization is a server virtualization method where the kernel of an operating system allows for multiple isolated user space instances, instead of just one. Such an instance is called container. Multiple isolated containers can run on a single host, using their own process and network space. The deployment of applications inside software containers can easily be automated, and identical runtime conditions for applications on any type of machine can be ensured. Containers and virtual machines (VMs) share many similarities but are fundamentally different because of the architecture. Containers run as lightweight processes within a host OS, whereas VMs depend on a hypervisor to emulate the architecture. Since there is no hypervisor involved, containers are faster, more efficient, and easier to manage.

CLC integrates Kubernetes into preassembled base images the nodes of a cluster. Kubernetes works in conjunction with Docker containers. Docker containers automate the deployment of applications inside software containers. Kubernetes is a cluster manager that can manage containerized applications across multiple hosts. It aggregates cluster resources into one large virtual machine, and provides basic mechanisms for deployment, maintenance, and scaling of applications. With CLC, Docker containers are automatically dispatched to cluster nodes.

While Docker provides the lifecycle management of containers, Kubernetes takes it to the next level by allowing for orchestration, managing clusters of containers, automating scaling and failover, and providing means of container interconnection.

For details, refer to *Cluster Orchestration* on page 13.

Automated Provisioning and Setup of Clusters

CLC provides an integrated, tested package including all components required to set up and operate a cluster. Cluster nodes are powered by CentOS Atomic Host, a light weight Linux operating system optimized to run containers. CLC provides preassembled nodes, which

automate the provisioning of clusters: The node image contains a ready-to-use Docker integration on CentOS.

For details, refer to *Cluster Management* on page 12.

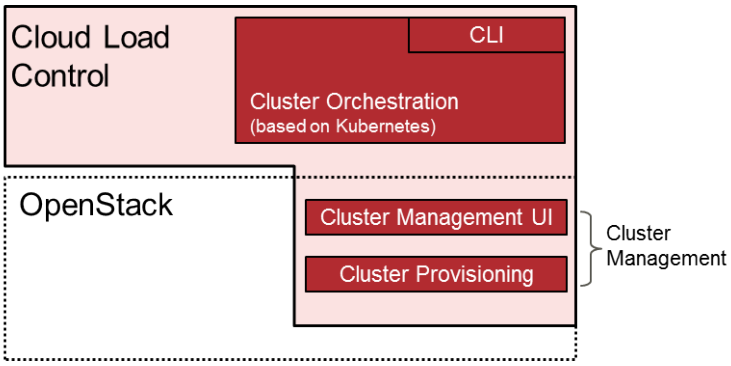
Seamless Integration with the OpenStack Dashboard

CLC is fully integrated with OpenStack and extends the OpenStack dashboard (Horizon) with an additional panel. A cluster operator can use this panel for requesting the automatic setup, provisioning, or deprovisioning of a cluster.

For details, refer to *Cluster Management* on page 12.

1.2 Components

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



OpenStack

OpenStack is the underlying platform technology of CLC. OpenStack is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a datacenter, all managed through a dashboard that gives administrators control while empowering their users to provision resources through a web interface. OpenStack allows for the deployment and management of Infrastructure as a Service (IaaS) platforms.

For details, refer to the *OpenStack* documentation.

Cluster Management

Cluster Management integrates closely with OpenStack. It consists of:

- **Cluster Provisioning**: Provides the functionality to automatically provision clusters based on resources provided by OpenStack and the subsequent setup of the Cluster Orchestration.
- **Cluster Management UI**: 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.

For details, refer to *Cluster Management* on page 12.

Cluster Orchestration

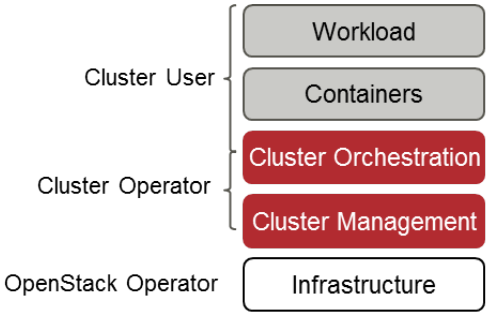
Cluster Orchestration based on Kubernetes is mostly independent from the underlying platform. It is the core of CLC.

The command-line interface (**CLI**) is used for maintenance and troubleshooting purposes, as well as for deploying workloads to the cluster nodes. In addition the status of a cluster can be retrieved so that optimizing cluster utilization is facilitated.

For details, refer to *Cluster Orchestration* on page 13.

1.3 Users and Roles

The CLC users can be grouped by their role. The following roles are distinguished:

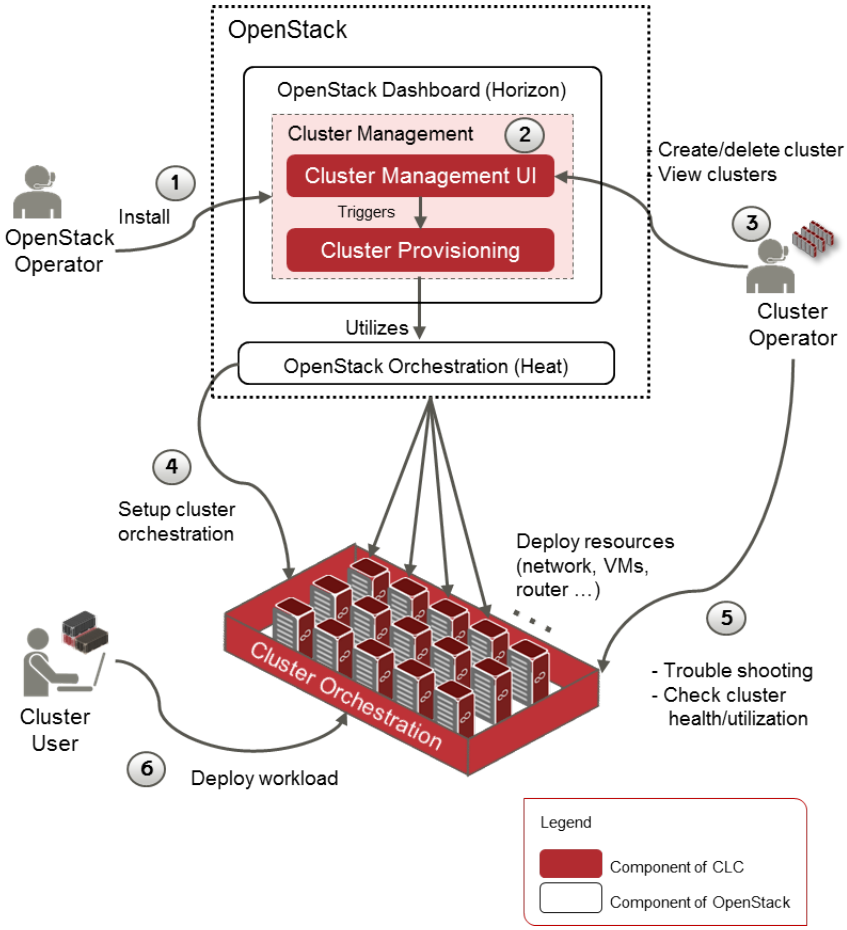


- The **cluster user** is responsible for developing and deploying applications on the cluster. He uses CLC for application deployment and ensuring workload mobility, scalability, and availability. For details, refer to the *Cluster Management Guide*.
- The **cluster operator** is responsible for managing clusters on an infrastructure and orchestration level. He uses CLC for cluster setup and management, and ensures high utilization of the cluster. For details, refer to the *Cluster Management Guide*.
- The **OpenStack operator** is responsible for administrating and maintaining the underlying OpenStack platform, and for ensuring the availability and quality of the OpenStack services. He installs CLC on an OpenStack platform. For details, refer to the *Installation Guide*.

Actual end users of services provided in the cluster do not interact with and are not aware of the underlying layers, such as containers or clusters. The end users benefit from CLC's performance and high availability.

1.4 Interaction of Components and Users

The following picture shows how the CLC components and users interact with each other:



1. The OpenStack operator installs CLC on top of OpenStack. The OpenStack operator can use an automated setup or follow manual step-by-step instructions as provided in the *Installation Guide*.

   In addition, 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 want to try this installation, please contact your Fujitsu Support organization.

2. The installation process automatically sets up the **Cluster Management** component which provides all artefacts and tools required for provisioning a cluster. Cluster Management utilizes the resources provided by the underlying OpenStack platform.

   Cluster Management comprises a **Cluster Management UI** and Cluster Provisioning which integrates seamlessly with the OpenStack dashboard (*Horizon*). It extends the dashboard with a panel for cluster management.

3. The Cluster operator uses the Cluster Management UI for creating and deleting clusters, as well as getting an overview of all existing clusters and their current state.

Cluster Provisioning is triggered when the cluster operator requests the creation of a new cluster. The operator provides the required information, for example, number of nodes to be provisioned, or access keys in the Cluster Management UI.

4. Based on the input values, instructions for provisioning the cluster are created and passed to the **OpenStack Orchestration** (**Heat**) system. 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** (Kubernetes) on top of the created resources.
5. The cluster operator uses the command-line interface (**CLI**) of the Cluster Orchestration for maintenance and troubleshooting purposes, as well as for retrieving the usage of the underlying resources, such as CPU and memory.
6. The cluster user deploys the actual workloads into the Docker containers using the CLI of the Cluster Orchestration.

2 Cluster Management

The Cluster Management component of CLC provides artefacts and tools for setting up and managing clusters. It utilizes the resources provided by the underlying OpenStack platform.

Cluster Management comprises a Cluster Management user interface (UI) and Cluster Provisioning.

Cluster Management UI

CLC comes with a plugin for the OpenStack dashboard (*Horizon*). It extends the dashboard with a panel the cluster operator can use for setting up and creating a new cluster by providing the following information:

- **Name** of the cluster.
- **Flavor**. Flavors are virtual hardware templates in OpenStack, defining sizes for RAM, disk, number of cores, and so on. The standard OpenStack installation provides several flavors. The default flavors are configurable by OpenStack administrators.
- **Node count**. Number of nodes to be created in the cluster to be provisioned. Servers that perform work are called nodes. Node servers communicate with the master server, configure the networking for containers, and run the actual workloads assigned to them. An Atomic host image with the CentOS operating system is provided by CLC as base image for the nodes.
- **Key pair for access & security**

  Cloud images work with public key authentication rather than conventional user name/ password authentication. When using the dashboard for the first time, a key pair must be created in OpenStack.
- **IP pool**. Range of floating IP addresses that can be assigned to the nodes in the cluster to be provisioned.

In addition, the cluster operator can use the Cluster Management UI for terminating and deprovisioning a cluster.

For details, refer to the *Cluster Management Guide*.

Cluster Provisioning

Cluster Provisioning is triggered via the Cluster Management UI when the cluster operator requests the creation of a cluster. Based on the input values, instructions for the provisioning of the cluster are created and passed to the OpenStack Orchestration (Heat) system.

CLC comes with predefined Heat orchestration templates (HOT). As part of the cluster setup, Heat is used to provision the VMs and configure the node operating system (CentOS). The preassembled base image for the nodes in the cluster is an *Atomic* host. This is a node with a lean operating system designed to run Docker containers, built from upstream CentOS. It provides all the benefits of upstream distribution, plus the ability to perform Atomic upgrades and rollbacks.

Heat is responsible for the actual creation of the resources as well as the installation and configuration of Kubernetes (Cluster Orchestration) on top of them.

3 Cluster Orchestration

Cluster Orchestration is the core of CLC. It utilizes *Kubernetes*, an open source project initiated by Google and developed by a large open source community. Below you find a description of the basic Kubernetes concepts.

Master Server

The controlling unit in a Kubernetes cluster is called the master server. It serves as the main management contact point for administrators, and provides many cluster-wide services for the node servers.

The master server runs a number of unique services that are used to manage the cluster's workload and direct communication across the system.

Node Server

In Kubernetes, servers that perform work are called nodes or node servers. Node servers communicate with the master server, configure the networking for containers, and run the actual workloads assigned to them.

Each individual node server is used to run *Docker*. The Docker service allows for running encapsulated application containers in a relatively isolated but lightweight operating environment. Each unit of work, at its basic level, is implemented as a series of containers that must be deployed.

In order to make services available to external parties, a proxy service runs on each node server. This service forwards requests to the correct containers, can do primitive load balancing, and is generally responsible for ensuring that the networking environment is predictable and accessible, but isolated.

Kubernetes Workloads

While containers are used to deploy applications, the workloads that define each type of work are specific to Kubernetes. Kubernetes applies the following concepts:

- **Clusters** are the compute resources on top of which the containers are build.
- **Pods** are a collocated group of Docker containers with shared volumes. They are the smallest deployable units that can be created, scheduled, and managed by the Cluster Orchestration. Pods can be created individually, but the usage of replication controllers is recommended.
- **Replication controllers** manage the lifecycle of pods. They ensure that a specific number of pods are running at a given time by creating or killing pods as required.
- **Services** provide a single, stable name and address for a set of pods. They act as basic load balancers.
- **Labels** are used to organize and select groups of objects based on key-value pairs.

Kubernetes Interface

Interaction with Kubernetes is possible via a command line interface (CLI). The cluster user deploys and manages workloads on the cluster with it. The cluster operator uses it for maintenance and troubleshooting purposes, as well as for retrieving the status of a cluster and its resources. Via the CLI, the operator can also access the logical components of a cluster, such as services or replication controllers.

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.