Hewlett Packard Enterprise

Red Hat

Reference Architecture

HPE Reference Configuration for Red Hat OpenShift Container Platform on HPE SimpliVity

Contents

Executive summary

Software development in the enterprise is undergoing rapid and widespread change. Application architectures are moving from monolithic and N-tier to cloud-native microservices, while the development process has transitioned from waterfall through agile to a DevOps focus. Meanwhile, deployments have moved from the data center to hosted environments and now the cloud (public, private, and hybrid), and release cycles have shrunk from quarterly to weekly, or even more frequently. To remain competitive, businesses require functionality to be delivered in a faster and more streamlined manner, while facing ever-increasing security threats.

Red Hat® OpenShift Container Platform 4.1 offers a consistent, self-managing enterprise Kubernetes platform that spans the hybrid cloud. Red Hat OpenShift 4 drives developer productivity while limiting operational complexities with native automation, providing a Kubernetes platform that pairs production-readiness with cloud-native innovation.

This Reference Configuration for Red Hat OpenShift Container Platform (OCP) on HPE SimpliVity is ideal for customers migrating legacy applications to containers, transitioning to a container DevOps development model or needing a hybrid environment to support container and non-containerized applications on a common VM platform. It provides a solution for IT operations, addressing the need for a production-ready environment that is easy to deploy and manage. It describes how to automate and accelerate the provisioning of the environment using a set of Ansible playbooks which, once configured, can deploy a cluster in thirty minutes. It also addresses post-deployment tasks such as enabling cluster logging, adding additional worker (compute) nodes, and backup and restore.

Target audience: This document is primarily aimed at technical individuals working in the operations side of the software pipeline, such as infrastructure architects, system administrators and infrastructure engineers, but anybody with an interest in automating the provisioning of virtual servers and containers may find this document useful.

Assumptions: A minimum understanding of concepts such as virtualization and containerization and also some knowledge around Linux® and VMware® technologies.

Solution overview

This Reference Configuration for Red Hat OpenShift Container Platform (OCP) on HPE SimpliVity is part of an overall solution from Hewlett Packard Enterprise. The solution consists of hardware and software, along with the HPE services options to help you deploy an enterprise-ready Kubernetes platform, quickly and efficiently. This solution takes HPE SimpliVity infrastructure and combines it with Red Hat's enterprise-grade OpenShift Container Platform and popular open source tools, and deployment and advisory services that are available from HPE Pointnext. The Deployment Guide provides detailed instructions on how to deploy the solution and is available at https://hewlettpackard.github.io/OpenShift-on-SimpliVity.

Kubernetes is an orchestration system for managing container-based applications. Kubernetes empowers developers to utilize new architectures like microservices and serverless that require developers to think about application operations in a way they may not have before. These software architectures can blur the lines between traditional development and application operations.

Red Hat OpenShift Container Platform expands enterprise Kubernetes with full-stack automated operations to manage hybrid cloud and multi-cloud deployments.

HPE SimpliVity is an enterprise-grade hyper-converged platform uniting best-in-class data services with the world's best-selling server.

Kubernetes overview

Kubernetes is an open-source container orchestrator project, founded by Google in 2014 and based on the internal distributed systems that support some of Google's most popular applications.

Containers

Containers are often viewed as lightweight virtual machines (VMs). While a VM has its own complete operating system sitting on top of a hypervisor, containers are sandboxes running directly on top of the host system's kernel and, as a result, are faster and less resource intensive.

Developers initially adopted containers as a means to package up their code, along with all of its dependencies and configuration details, to run it anywhere - public, private or hybrid cloud. By simplifying the development environment, container technology allowed developers to run multiple versions of their own applications and 3rd party software on a single workstation without annoying conflicts. As a result, containers became a common, standardized building block for software development and led to the demise of the "it works on my machine" scenario.

Containers have been around for a long time in the Linux world, but Docker popularized them by making them easy and efficient to use and by providing a public registry of standardized container images for 3rd party software. In the past, container technology had been perceived to be prone to security vulnerabilities, in particular to "breakout" where malicious code could escape the sandbox and access sensitive information on the host. Over the years, extensive work has been done to reduce the attack surface and to limit the blast-radius, should any attack succeed. As a result, running applications on containers can now significantly reduce the impact of any attack due to the underlying protections provided.

Containers also facilitated the adoption of microservices architectures where, instead of developing single monolithic programs, applications are split up into a set of independent services that communicate with each other via well-defined interfaces (APIs). As a result, the container has now become the standardized unit for software development for packaging, composition, deployment, scaling and re-use. However, to deploy and maintain a reliable distributed system using all these containers, another layer of management software is required and that is role of the container orchestrator.

Container orchestration

A container orchestrator is a piece of software that attempts to automate the operations that would traditionally be performed by a system administrator. This includes:

- Scaling applications up and down, depending on demand
- Load balancing across containers
- Restarting individual containers that fail
- Replacing and rescheduling containers when an underlying host node dies
- Managing compute, memory, network and storage resources
- Optimizing resource utilization
- Automating the roll-out and rollback of deployments
- Allowing services to discover other services in the system
- Monitoring and centralized logging

The use of an orchestrator typically results in increased container density, leading to improved overall utilization of resources. In addition, the average lifetime of a container also decreases significantly as the orchestrator restarts, removes or relocates containers when auto-scaling or when node failure occurs.

Kubernetes

A number of proprietary container orchestration systems have been available, including Docker Swarm and Mesosphere DC/OS. However, the DevOps community rapidly converged to make Kubernetes the de-facto standard and most commercial offerings have now pivoted to include Kubernetes as part of their offerings. It should be noted that Kubernetes also underpins offerings from all the main cloud providers such as Amazon's Elastic Container Service for Kubernetes (EKS), Microsoft's® Azure® Kubernetes Service (AKS) and Google's own Kubernetes Engine (GKE). This ability to support on-premises, public cloud and hybrid deployments using a single technology and avoiding vendor lock-in, helps further confirm the grip Kubernetes has on the DevOps mindset.

For more information on Kubernetes, refer to https://kubernetes.io/.

Red Hat OpenShift Container Platform overview

Red Hat OpenShift Container Platform 4 is a consistent, managed Kubernetes experience which seamlessly connects workloads between on-premise data centers and public cloud footprints. Red Hat OpenShift 4 drives developer productivity while limiting operational complexities with built-in automation.

Trusted enterprise Kubernetes

Red Hat OpenShift Container Platform is certified, conformant Kubernetes, as validated by the Cloud Native Computing Foundation (CNCF). It is the only enterprise Kubernetes offering built on the backbone of the world's leading enterprise Linux platform backed by the open source expertise, compatible ecosystem, and leadership of Red Hat. As a top Kubernetes community contributor, Red Hat refines Kubernetes for enterprises in Red Hat OpenShift 4, providing a codebase that is hardened and more secure while retaining key innovations from upstream communities.

Red Hat Enterprise Linux CoreOS

To provide more flexible deployment footprint, while still maintaining enhanced security and stability, Red Hat OpenShift 4 introduces Red Hat Enterprise Linux CoreOS, an OpenShift-specific embedded variant of Red Hat Enterprise Linux. Red Hat Enterprise Linux CoreOS provides expanded choice for enterprises in deploying enterprise-grade Kubernetes, offering a lightweight, fully immutable, container-optimized Linux OS distribution. In this variant, security features and stability are still paramount, with automated updates managed by Kubernetes and enabled by OpenShift with the push of a button. This helps to reduce maintenance and improve business productivity.

Kubernetes operators

Kubernetes operators support the complete automation for an application, by handling requirements such as updates, backups, recovery from failure and scaling (up and down) without the need for human intervention. Operators extend Kubernetes functionality in a standardized fashion, improving resiliency while reducing the load on operations staff. In particular, in multi-cloud or hybrid cloud scenarios, they can shield developers of cloud-native applications from variations across deployment environments.

Red Hat OpenShift Container Platform makes extensive use of Kubernetes operators to encode domain knowledge to correctly scale, upgrade, and reconfigure your cluster and workloads, while protecting against data loss or unavailability.

- Operators are built into OpenShift, so Kubernetes and cluster services are always up-to-date.
- Embedded OperatorHub provides a discovery marketplace for independent software vendor (ISV) operators, validated to run on OpenShift.

Full-stack automated operations

Once the cluster and applications are deployed, life-cycle management for these components, consoles for operators and developers, and security throughout the entire life cycle become critical. Red Hat OpenShift Container Platform offers automated installation, upgrades, and life-cycle management for every part of your container stack including the CoreOS operating system, Kubernetes, and cluster services and applications. The result is a more secure, always up-to-date Kubernetes application platform, without the headaches of manual and serial upgrades, or downtime.

Developer productivity

Developers can quickly and easily create applications on demand from the tools they use most, while operations retains full control over the entire environment. Integrated CI/CD pipelines let developers reduce manual deployment work and deploy higher quality software for continuous integration and automated tests. Support for leading-edge tools such as OpenShift Service Mesh and Knative serverless help drive developer productivity and innovation.

An introduction to the Red Hat OpenShift Container Platform architecture is available at https://docs.openshift.com/container-platform/4.1/architecture/architecture.html.

HPE SimpliVity overview

HPE SimpliVity is an enterprise-grade hyper-converged platform uniting best-in-class data services with the world's best-selling server.

Rapid proliferation of applications and the increasing cost of maintaining legacy infrastructure causes significant IT challenges for many organizations. With HPE SimpliVity, you can streamline and enable IT operations at a fraction of the cost of traditional and public cloud solutions, by combining your IT infrastructure and advanced data services into a single, integrated solution. HPE SimpliVity is a powerful, simple, and efficient hyperconverged platform that joins best-in-class data services with the world’s best-selling server and offers the industry’s most complete guarantee.

For more information about HPE SimpliVity, refer to https://www.hpe.com/us/en/integrated-systems/simplivity.html.

Solution configuration

The solution deploys an OCP cluster and a number of supporting nodes, as shown in Figure 1.

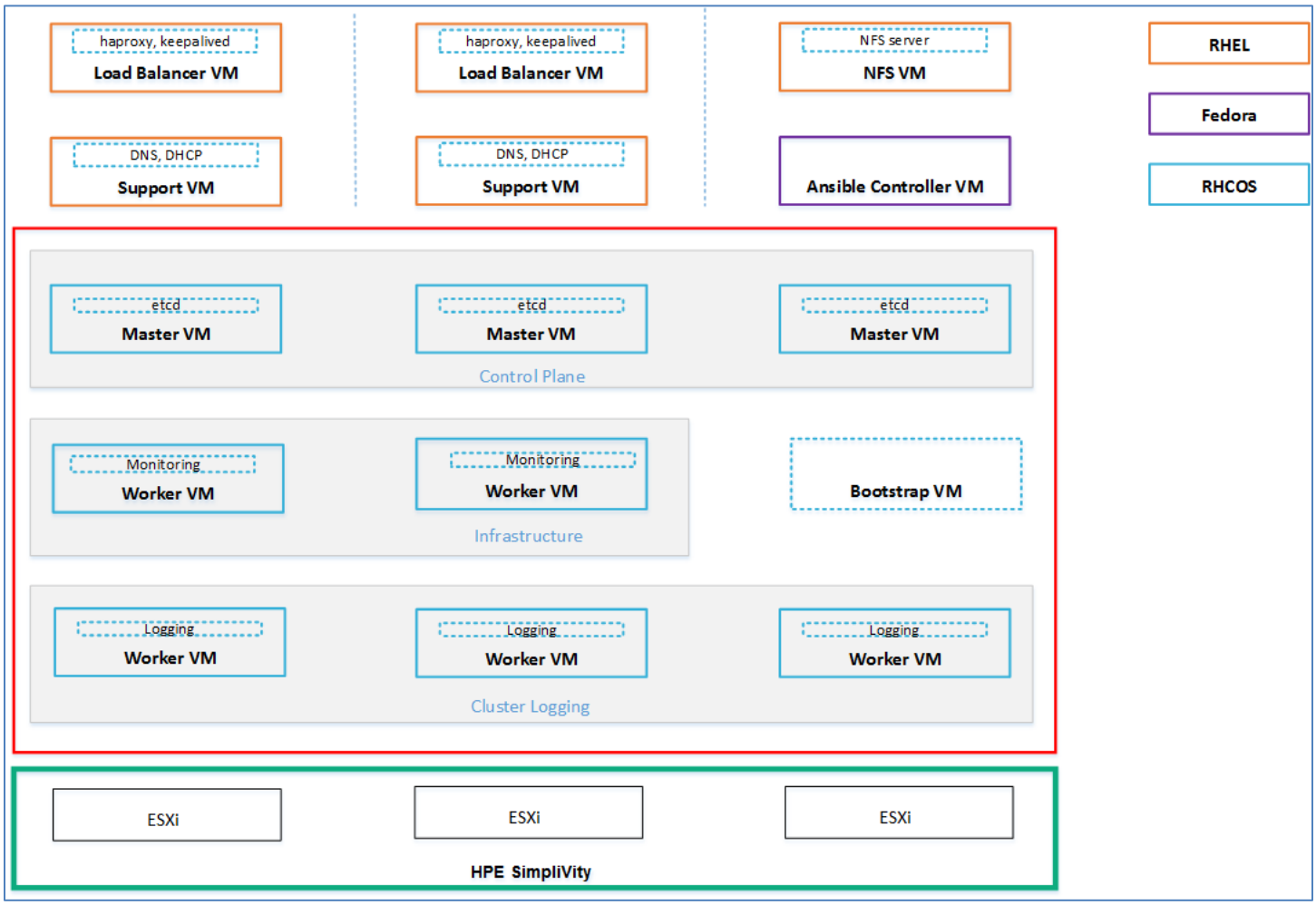

Figure 1. Solution overview, including optional cluster logging stack configured after initial deployment

Cluster nodes

The OpenShift Container Platform cluster consists of two distinct types of nodes:

- **Master nodes:** A minimum of three master nodes are required for the control plane.
- **CoreOS worker nodes:** A minimum of two CoreOS worker nodes must be specified for the infrastructure components in the initial cluster deployment. The default router, image registry and monitoring service are initially deployed on these two worker nodes, along with any applications you subsequently deploy. The worker nodes will be distributed randomly, so they may not be spread across the ESXi hosts as depicted in the figure.

Supporting nodes

In addition to the cluster nodes, a number of supporting nodes are also deployed. They are:

- **Bootstrap**: A single bootstrap machine is used to deploy the Red Hat OpenShift Container Platform cluster on the three master nodes. You can safely remove the bootstrap machine from vCenter® after the cluster has deployed.

- **DNS and DHCP**: The solution provisions DNS and DHCP services on one or two support nodes. For high availability (HA) purposes in a production environment, you should configure two support nodes. For proof of concept deployments, one node should be sufficient.
- **Load balancers**: The playbooks can be used to deploy one or two load balancers depending on your high availability requirements. Alternatively, you can configure your own load balancers. For more information, see the section on Load balancer configuration in the Deployment Guide.
- **NFS server**: The solution deploys a single NFS server to provide persistent storage as required by the OCP image registry.

Cluster logging

Cluster logging is very resource intensive and not deployed by default. A separate playbook is provided to configure cluster logging after the initial deployment. The playbook does not create anti-affinity rules for the logging VMs, so the actual distribution may differ from that depicted in the solution overview diagram.

Red Hat Enterprise Linux 7.6 worker nodes

RHEL 7.6 worker nodes can be deployed as an alternative to CoreOS worker nodes, but can only be added to the OCP cluster after the cluster has been deployed. For more information, see the section on Deploying RHEL worker nodes in the Deployment Guide.

Solution sizing

By default, the Ansible playbooks create VMs according to the following recommended specifications. You can override these recommended values in the configuration files, as detailed in the section Configuring the solution in the Deployment Guide.

Bootstrap node

A single bootstrap node is required to assist in the OCP deployment, with the minimum sizing shown in Table 1. You can delete the bootstrap VM after the cluster has been successfully deployed.

Table 1. Sizing for bootstrap node

VM	Number	OS	Sizing	Comments
Bootstrap	1	CoreOS	4x vCPU 16GB RAM 120GB disk space	This is the Red Hat minimum requirement

Supporting nodes

The two supporting nodes are deployed by default, providing DHCP and DNS services for the cluster, with the minimum sizing shown in Table 2. Two load balancer VMs and an NFS VM are also configured.

Table 2. Sizing for supporting nodes

VM	Number	OS	Sizing	Comments
Support	2	RHEL 7.6	2x vCPU 4GB RAM 60GB disk space	Providing DHCP and DNS services on the internal VLAN. You can configure one (no HA)
Load balancers	2	RHEL 7.6	2x vCPU 4GB RAM 60GB disk space	Two load balancers are deployed by default. You can configure one (no HA) or 0 where you use your own existing load balancers
NFS	1	RHEL 7.6	2x vCPU 4GB RAM 60GB disk space	Required for persistent storage for the OpenShift Registry

OCP cluster nodes

By default, 3 master nodes are deployed for high availability while a minimum of 2 worker nodes are required for the initial deployment. The minimum sizing for master and worker nodes is shown in Table 3.

Table 3. Sizing for master and worker nodes

VM	Number	OS	Sizing	Comments
Masters	3	CoreOS	4x vCPU 16GB RAM 120GB disk space	This is the Red Hat minimum requirement
Workers	2	CoreOS	2x vCPU 16GB RAM 120GB disk space	This is the Red Hat minimum requirement

Similar sizing requirements will apply for any worker nodes (CoreOS or RHEL) added to the cluster after the initial deployment.

OCP infrastructure components

The following OpenShift Container Platform components are infrastructure components:

- Kubernetes and OpenShift Container Platform control plane services that run on masters.
- The default router.
- The container image registry.
- The cluster metrics collection, or monitoring service.
- Cluster aggregated logging.
- Service brokers.

Any node that runs any other container, pod, or component is a worker node that your Red Hat OpenShift subscription must cover.

The default router, image registry and monitoring service are initially deployed on the two worker nodes. These can be rescheduled onto specific infrastructure nodes as outlined in the section Post deployment tasks in the Deployment Guide.

Cluster Logging

Cluster logging is very resource intensive, so the sizing requirements, shown in Table 4, are increased compared to other nodes in the cluster.

Table 4. Sizing for logging nodes

VM	Number	OS	Sizing	Comments
Logging	3	CoreOS	8x vCPU 32GB RAM 120GB disk space	This is the Red Hat recommended requirement

Cluster logging is not deployed by default. Instructions for deploying the logging stack and scheduling it on specific nodes with the required resources is described in the section Deploying cluster logging in the Deployment Guide.

High availability

Anti-affinity rules are created to ensure that most VMs performing the same role are kept apart and run on different underlying ESXi hosts.

Control plane

The default configuration for the solution is three master nodes running on three separate ESXi hosts. The control plane VMs are guaranteed to run on different ESXi hosts through the use of an anti-affinity rule named:

```
{{cluster_name}}-master-anti-affinity-rule-001
```

where `cluster_name` is the name of your cluster, as defined in the `group_vars/all/vars.yml` file.

Deploying two support nodes

You can configure the internal DNS and DHCP services to run on two virtual machines to provide redundancy. These two VMs are guaranteed to run on two different ESXi hosts through the use of an anti-affinity rule named:

```
{{cluster_name}}-support-anti-affinity-rule-001
```

where `cluster_name` is the name of your cluster, as defined in the `group_vars/all/vars.yml` file.

Deploying two load balancers

You can configure the playbooks to deploy two load balancers in an active-active configuration to provide high availability access. These nodes run `keepalived` and HAproxy. The load balancers are hosted on two VMs that are guaranteed to run on two different ESXi host using an anti-affinity rule named:

```
{{cluster_name}}-loadbalancer-anti-affinity-rule-001
```

where `cluster_name` is the name of your cluster, as defined in the `group_vars/all/vars.yml` file.

Deploying cluster logging

Cluster logging is not deployed by default. Deployment of the logging stack and how to schedule it on nodes with the required resources is described in the section Deploying cluster logging in the Deployment Guide.

Ansible playbooks

The OpenShift-on-SimpliVity Ansible playbooks are available to download at https://github.com/HewlettPackard/OpenShift-on-SimpliVity. By default, the playbooks are configured to initially deploy three master nodes and two worker nodes. This is the minimal starter configuration recommended by Hewlett Packard Enterprise and Red Hat for production.

Figure 2. Solution overview

Figure 2 provides a solution overview and below are the steps used to deploy the solution.

- Create the Fedora Ansible host.
- Prepare the RHEL 7.6 VM template.
- Configure the required Ansible parameters.
- Run the Ansible playbooks for initial deployment.
- Perform post-deployment tasks and validation.
- Add CoreOS and RHEL 7.6 worker nodes to the cluster.
- Deploy cluster logging.
- Backup and restore your cluster.

Deploying HPE SimpliVity hardware is specific to your environment and is not covered in this document. HPE SimpliVity documentation is available in the Hewlett Packard Enterprise Support Center.

Solution components

Hardware

The hardware is comprised of three HPE SimpliVity 380 Gen10 servers. Hewlett Packard Enterprise recommends dual socket HPE SimpliVity systems with at least 14 CPU cores per socket (28 total cores per system) for optimal performance and support during HA failover scenarios. Since the HPE SimpliVity technology relies on VMware virtualization, the servers are managed using vCenter.

For more details, see the Bill of Materials in Appendix A: Bill of Materials.

Software

The software components used in this Reference Configuration are listed below.

Table 5. Third-party software

Component	Version
Ansible	2.8.1
Red Hat OpenShift Container Platform	4.1.* Tested using 4.1.16
Red Hat CoreOS	4.1.0
Red Hat Enterprise Linux	7.6
VMware	vCenter 6.7 U2

Table 6. HPE Software

Component	Version
HPE SimpliVity OmniStack	3.7.10

About Ansible

Ansible is an open-source automation engine that automates software provisioning, configuration management and application deployment.

As with most configuration management software, Ansible has two types of servers: the controlling machine and the nodes. A single controlling machine orchestrates the nodes by deploying modules to the Linux nodes over SSH. The modules are temporarily stored on the nodes and communicate with the controlling machine through a JSON protocol over the standard output. When Ansible is not managing nodes, it does not consume resources because no daemons or programs are executing for Ansible in the background. Ansible uses one or more inventory files to manage the configuration of the multiple nodes in the system.

For more information about Ansible, refer to http://docs.ansible.com.

Application software

Prometheus and Grafana

OpenShift Container Platform includes a pre-configured, pre-installed, and self-updating monitoring stack that is based on the Prometheus open source project and its wider eco-system. It provides monitoring of cluster components and includes a set of alerts to immediately notify the cluster administrator about any occurring problems, as well as a set of Grafana dashboards. The cluster monitoring stack is only supported for monitoring OpenShift Container Platform clusters. If you wish to monitor your own application workloads, you should deploy a separate Prometheus instance.

All the components of the monitoring stack are monitored by the stack and are automatically updated when OpenShift Container Platform is updated. The monitoring stack also monitors cluster components including:

- Etcd

- Kubernetes apiserver, controller manager, scheduler
- OpenShift apiserver, controller manager, Lifecycle Manager
- CoreDNS and HAProxy
- Image registry
- Elasticsearch and Fluentd (if cluster logging is installed)

For more information on the monitoring stack, see the documentation at https://docs.openshift.com/container-platform/4.1/monitoring/cluster-monitoring/about-cluster-monitoring.html.

Elasticsearch, Fluentd and Kibana

Hewlett Packard Enterprise provides an additional playbook to deploy the integrated cluster logging stack, which aggregates logs for a range of OpenShift Container Platform services.

The cluster logging components are based upon Elasticsearch, Fluentd, and Kibana (EFK). The collector, Fluentd, is deployed to each node in the OpenShift Container Platform cluster. It collects all node and container logs and writes them to Elasticsearch (ES). Kibana is the centralized, web UI where users and administrators can create rich visualizations and dashboards with the aggregated data.

Fluentd

OpenShift Container Platform uses Fluentd to collect data about your cluster. Fluentd is deployed as a DaemonSet in OpenShift Container Platform, which deploys pods to each OpenShift Container Platform node. Fluentd uses `journald` as the system log source. These are log messages from the operating system, the container runtime, and OpenShift Container Platform.

Elasticsearch

OpenShift Container Platform uses Elasticsearch (ES) to organize the log data from Fluentd into datastores, or indices. Elasticsearch subdivides each index into multiple pieces called shards, which it spreads across a set of Elasticsearch nodes in an Elasticsearch cluster. You can configure Elasticsearch to make copies of the shards, called replicas. Elasticsearch also spreads these replicas across the Elasticsearch nodes.

Kibana

OpenShift Container Platform uses Kibana to display the log data collected by Fluentd and indexed by Elasticsearch. Kibana is a browser-based console interface to query, discover, and visualize your Elasticsearch data through histograms, line graphs, pie charts, heat maps, built-in geospatial support, and other visualizations.

Preparing the environment

Verify prerequisites

Before you start deployment, you must assemble the information required to assign values for each variable used by the playbooks. The variables are fully documented in the section Configuring the solution in the Deployment Guide. A brief overview of the information required is presented in Table 7.

Table 7. Overview of prerequisites

Component	Details	Comments
Virtual Infrastructure	ESXi cluster of three machines	If you want HA, you need three machines in the cluster and you need to deploy 3 masters, one per machine.
	vCenter 6.7U2	The FQDN of your vCenter server and the name of the data center are required. You will also need administrator credentials in order to create templates and spin up virtual machines.

Component	Details	Comments
Proxy-free VLAN with access to Internet (to pull Red Hat artifacts)	Portgroup connected to all machines in your ESX cluster	The playbooks install DHCP services on this VLAN so no other DHCP service should be running on this VLAN. You will need one IP address for each VM configured in the Ansible inventory.
Routed subnet for use on the above VLAN		
Frontend Network / VLAN		External IP address for each load balancer plus one for the frontend VIP
NTP Services	IP addresses of time servers (NTP)	Time services must be configured in your environment. The deployed solution uses certificates that are time-sensitive.
DNS Services	IP addresses of DNS servers	The DNS services deployed by the solution forwards unresolved requests to these DNS servers
Red Hat Network subscription	Organization ID and authorization key Alternatively a username and password.	The subscription must contain valid licenses for both Red Hat Enterprise Linux 7.6 and OpenShift Container Platform 4

Create the Ansible node on Fedora

It is recommended that you deploy the Ansible controller on Fedora, to take advantage of the built-in support for Python 3. The solution requires Fedora 29 and Ansible 2.8.1 and the node should be directly connected to the proxy-free VLAN. It may be possible to run the playbooks on other operating systems including RHEL and CentOS but the playbooks have only been tested on Fedora. In addition to Python 3 and Ansible 2.8.1, the following packages must be installed: `python3-netaddr python3-requests python3-pyvmomi python3-pip`.

The playbooks should be run from a non-privileged account. Hewlett Packard Enterprise recommends that you use an account named `core` as this will ensure consistency with deployed CoreOS machines that have a built-in account `core`. In addition, any non-CoreOS VMs created by the playbooks will have a user account created with the same name as the user on the Ansible controller node that ran the playbooks. The detailed instructions for setting up the Ansible controller node are available in the section Create the Ansible node on Fedora in the Deployment Guide.

Create the Red Hat Linux template

To create the Red Hat Linux VM template that will be used as the base image for all non-RHCOS VM nodes, you first create a Virtual Machine, install the base RHEL 7.6 OS and then convert the VM to a VM template. Any additional software installation and/or system configuration is performed subsequently using Ansible, rather than in the template itself.

As the creation of the template is a one-time task, this procedure has not been automated. The steps required to manually create a VM template are detailed in the section Create the Red Hat Linux template in the Deployment Guide.

OpenShift artifacts

Architecture

An introduction to the Red Hat OpenShift Container Platform architecture is available at https://docs.openshift.com/container-platform/4.1/architecture/architecture.html.

Installation

Red Hat OpenShift Container Platform supports two types of installation programs:

- Installer-Provisioned Infrastructure (IPI): The installation program is responsible for deploying and maintaining the underlying infrastructure, as well as the cluster itself. An example of this is the deployment to Amazon Web Services (AWS).
- User-Provisioned Infrastructure (UPI): The user is responsible for setting up and maintaining the underlying infrastructure, and the installation program deploys the cluster on top of this. An example of this style of installation is deployment to bare metal or to VMware vSphere®.

As this solution runs on HPE SimpliVity, it follows the methodology for User-Provisioned Infrastructure, and it helps you to deploy the underlying VMs for the cluster itself and for supporting machines.

More information on the general installation process is available at https://docs.openshift.com/container-platform/4.1/architecture/architecture-installation.html.

Documentation specific to installation on a vSphere cluster is at https://docs.openshift.com/container-platform/4.1/installing/installing_vsphere/installing-vsphere.html.

Release notes

Red Hat OpenShift Container Platform provides regular updates, so it is important to follow the release details provided and to upgrade regularly. The OpenShift release notes for 4.1 are available at https://docs.openshift.com/container-platform/4.1/release_notes/ocp-4-1-release-notes.html.

Red Hat CoreOS

Download the Red Hat CoreOS OVA for OCP 4.1 from https://mirror.openshift.com/pub/openshift-v4/dependencies/rhcos/4.1/latest/rhcos-4.1.0-x86_64-vmware.ova.

This downloaded OVA will be used for the master and worker nodes in the initial cluster deployment. As part of the installation process, templates will be created from the master and worker OVAs.

Clients

The latest version of the client software is available at https://mirror.openshift.com/pub/openshift-v4/clients/ocp/latest/.

At the time of writing, the latest version of OpenShift is `4.1.16` and this is reflected in the sample download instructions below. If you want to use a newer version of the software, you must modify the URLs to match the version.

The `openshift-install` program used by the playbooks to install the cluster can be downloaded from https://mirror.openshift.com/pub/openshift-v4/clients/ocp/latest/openshift-install-linux-4.1.16.tar.gz.

Two client programs are used to interact with a deployed cluster - `oc` and `kubectl`. These programs are available in a single file at https://mirror.openshift.com/pub/openshift-v4/clients/ocp/latest/openshift-client-linux-4.1.16.tar.gz.

Pull secret

From the OpenShift Infrastructure Providers page, download your installation pull secret. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components. The pull secret will be used in your configuration using the variable `vault.pull_secret`, in the `group_vars/all/vault.yml` file.

Configuring and deploying the solution

On the Ansible controller node, retrieve the latest version of the playbooks using Git.

```
$ cd ~
$ git clone https://github.com/HewlettPackard/OpenShift-on-SimpliVity.git
```

Change to the directory that was created using the `git clone` command:

```
$ cd ~/OpenShift-on-SimpliVity
```

You now need to prepare the configuration to match your own environment, prior to deploying OpenShift. To do so, you will need to create and populate a number of files including:

- `hosts` - the inventory file
- `group_vars/all/vars.yml` - the group variables file used by all playbooks and roles
- `group_vars/all/vault.yml` - the global vault file containing sensitive information that needs to be protected

The Deployment Guide provides details of the various configuration options. To get started, corresponding sample files that you can use as a basis for your configuration are provided as part of the download:

- `hosts.sample`
- `group_vars/all/vars.sample`
- `group_vars/all/vault.sample`

Deploying the solution

Once you have configured all of the variables and vault parameters to match your environment, run the playbook `site.yml` to perform the initial cluster deployment:

```
$ cd ~/OpenShift-on-SimpliVity
$ ansible-playbook -i hosts site.yml --vault-password-file .vault_pass
```

Refer to the Deployment Guide for some common post-deployment tasks and validation. This includes:

- Logging into the OCP cluster for the first time
- Deploying a sample application
- Configuring external routes
- Exposing the image registry
- Integrating with LDAP
- Using labels to schedule workloads on specific nodes

Configuring storage

There are two types of storage supported in this release:

- vSphere Cloud Provider
- NFS

Persistent storage

Persistent storage is required by a number of cluster components and also may be required by any stateful application workloads running on your cluster. Among the built-in OCP components that can take advantage of persistent storage are:

- Image registry
- Prometheus, part of the monitoring stack
- Elasticsearch and Alertmanager, part of the logging stack

By default, the OpenShift installer configures a default storage class `thin` which uses the vSphere Cloud Provider.

Image registry

The default vSphere Cloud Provider does not support the `ReadWriteMany` access mode required by the Image Registry. As a result, the playbook for initial cluster deployment, `site.yml`, deploys an NFS virtual machine. A number of NFS shares are created and exported by the playbooks, and the Image Registry uses one of these NFS shares.

The number of NFS shares created by the playbooks can be customized using the variable `num_nfs_shares` in the configuration file `group_vars/all/vars.yml`. Only one share is required by the Image Registry service.

Scaling up the cluster with additional worker nodes

Once your OpenShift cluster has been successfully deployed, you can scale up the cluster with additional compute resources. These worker nodes are typically used for scheduling application workloads, so the number and size of the worker nodes will depend on the applications you wish to deploy in your OCP cluster. With OCP 4.1, two types of worker nodes are supported:

- Red Hat Core OS

- Red Hat Enterprise Linux (RHEL) 7.6

Deploying CoreOS worker nodes

As part of the initial deployment of the OpenShift cluster, two Red Hat Core OS workers are configured by default. The sample configuration for the initial cluster is three CoreOS master nodes and two CoreOS worker nodes.

To deploy additional CoreOS worker nodes, add new entries to the `[rhcos_worker]` group in your `hosts` file and run the Ansible playbook `playbooks/scale.yml`

```
$ cd ~/OpenShift-on-SimpliVity
$ ansible-playbook -i hosts playbooks/scale.yml --vault-password-file .vault_pass
```

The playbook will provision new VMs for the nodes, and these should automatically join the cluster after a few minutes. You can observe the nodes joining the cluster via the `oc get nodes` command. Initially, the nodes will report as not ready but after a minute or two, the status should change to `Ready`.

Removing CoreOS nodes

If you want to reduce the number of CoreOS worker nodes (for example, you may have replaced them with RHEL worker nodes), you need to use the following procedure. For each node that you want to remove, you must:

- Mark the node as unschedulable.
- Drain all the pods from the node.
- Delete the node.

Finally, remove the nodes from your load balancers and delete the VMs.

For more information on removing CoreOS nodes, see the documentation at https://docs.openshift.com/container-platform/4.1/machine_management/adding-rhel-compute.html#rhel-compute-requirements_adding-rhel-compute.

Deploying RHEL worker nodes

Red Hat Enterprise Linux (RHEL) worker nodes can only be deployed after the OpenShift control plane has been successfully deployed and once the OCP cluster is up and running.

Note

If you choose to use RHEL compute machines in your cluster, you take responsibility for all operating system life cycle management and maintenance, including performing system updates, applying patches, and completing all other required tasks.

Note

Only RHEL 7.6 is supported in OpenShift Container Platform 4.1. You must not upgrade your compute machines to RHEL 8.

System requirements for RHEL compute nodes

RHEL compute nodes in your OpenShift Container Platform environment must meet the following minimum hardware specifications and system-level requirements.

- You must have active OpenShift Container Platform and Red Hat Enterprise Linux Server subscriptions associated with your Red Hat Network account.
- Production environments must provide compute resources to support your expected application workloads. As an OpenShift Container Platform cluster administrator, you must calculate the resource requirements for your expected workload and add about 10 percent for overhead. For production environments, be sure to allocate enough resources so that your OCP cluster can survive a node failure event without affecting your running applications.

Running the playbooks

There are a number of steps for deploying RHEL worker nodes:

- Download Red Hat Ansible playbooks to the folder `~/openshift-ansible` folder
- Run the HPE playbook playbooks/scale.yml in the `~/Openshift-on-SimpliVity` folder to provision the RHEL VMs
- Run a Red Hat Ansible playbook to add the RHEL nodes to the OCP cluster.

Download Red Hat Ansible playbooks

The Red Hat OpenShift Ansible playbooks are available at https://github.com/openshift/openshift-ansible. For OpenShift 4.x, this repository only provides playbooks necessary for scaling up or upgrading RHEL hosts in an existing 4.x cluster.

Note

The required Red Hat `openshift-ansible` playbooks change regularly. Hewlett Packard Enterprise has no control over these playbooks and therefore recommends that you use a specific version of the playbooks rather than the latest version on the master branch. At the time of writing, the `4.1.11-201908060314` version of the Red Hat `openshift-ansible` playbooks have been tested and certified with this solution.

Deploying a RHEL worker

Detailed instructions are provided in the section Deploying RHEL worker nodes in the Deployment Guide. The following is a brief overview of the steps required.

- Clone the `4.1.11-201908060314` branch of the `openshift-ansible` playbooks into the home directory of the `core` user on the Ansible controller node.
- Modify your `hosts` inventory file to add one or more entries in the `[rhel_worker]` group.
- Run the scaling playbook from this solution `~/OpenShift-on-SimpliVity/playbooks/scale.yml`
- Run the Red Hat Ansible scaling playbook `~/openshift-ansible/playbooks/scaleup.yml`
- Verify that the newly created RHEL worker nodes have joined the cluster. It can take 10 to 15 minutes to deploy two Red Hat Enterprise Linux worker nodes.

Deploying cluster logging

You can deploy cluster logging to aggregate logs for a range of OpenShift Container Platform services.

The built-in cluster logging components are based upon Elasticsearch, Fluentd, and Kibana (EFK). The collector, Fluentd, is deployed to each node in the OpenShift Container Platform cluster. It collects all node and container logs and writes them to Elasticsearch (ES). Kibana is the centralized, web UI where users and administrators can create rich visualizations and dashboards with the aggregated data.

About cluster logging components

There are four different types of cluster logging components:

- **logStore**: This is where the logs will be stored. The current implementation is Elasticsearch.
- **collection**: This is the component that collects logs from the node, formats them, and stores them in the logStore. The current implementation is Fluentd.
- **visualization**: This is the UI component used to view logs, graphs, charts, and so forth. The current implementation is Kibana.
- **curation**: This is the component that trims logs by age. The current implementation is Curator.

For more information about the OpenShift Cluster Logging facility, refer to documentation at https://docs.openshift.com/container-platform/4.1/logging/efk-logging.html.

Deploying worker nodes for cluster logging

Elasticsearch is a memory-intensive application. Each Elasticsearch node needs 16GB of memory and extra CPU resources. The initial set of deployed OpenShift Container Platform worker nodes might not be large enough to support the Elasticsearch cluster. You must add additional nodes to the OpenShift Container Platform cluster to run with the recommended or higher memory and CPU requirements. Each Elasticsearch node can operate with a lower memory setting though this is not recommended for production deployments.

You should add new worker nodes to your cluster to facilitate hosting the logging stack, setting the `cpu` and `ram` attributes to sufficiently large values for a production environment. In your `hosts` file, add new entries in the `[rhcos_worker]` group:

```
[rhcos_worker]
hpe-worker0   ansible_host=10.15.155.213
hpe-worker1   ansible_host=10.15.155.214
hpe-worker2   ansible_host=10.15.155.215  cpus=8 ram=32768  # Larger worker node for EFK
hpe-worker3   ansible_host=10.15.155.216  cpus=8 ram=32768  # Larger worker node for EFK
hpe-worker4   ansible_host=10.15.155.217  cpus=8 ram=32768  # Larger worker node for EFK
```

In the above example, each of these large CoreOS worker nodes will be allocated 8 virtual CPU cores and 32GB of RAM. These values override the default limits of 4 virtual CPU cores and 16GB RAM defined in the `group_vars/worker.yml` file.

Deploy the additional, large worker nodes using the procedure described in the section Deploying CoreOS worker nodes in the Deployment Guide.

Playbooks for cluster logging

After making the customizations appropriate to your environment, deploy the EFK stack by changing to the directory where you cloned the OpenShift-on-SimpliVity repository and running the playbook `playbooks/efk.yml`:

```
$ cd ~/OpenShift-on-SimpliVity
$ ansible-playbook -i hosts playbooks/efk.yml
```

The playbook takes approximately 1-2 minutes to complete. However, it may take several additional minutes for the various cluster logging components to successfully deploy to the OpenShift Container Platform cluster.

Validating the cluster logging deployment

To verify the cluster logging deployment, in the web console:

- Switch to the `Workloads -> Pods` page
- Select the `openshift-logging` project

You should see several pods for cluster logging, Elasticsearch, Fluentd, and Kibana. Alternatively, you can use the `oc client` instead:

```
$ oc get pod -n openshift-logging
```

Access the Kibana Dashboard

Once the cluster logging instance has deployed successfully, a new entry called `Logging` will appear under the `Monitoring` tab of the OpenShift Container Platform dashboard. Selecting the `Logging` entry will launch the Kibana dashboard in a separate browser tab.

The Kibana dashboard can also be accessed directly at: `https://kibana-openshift-logging.apps.<<cluster_name>>.<<domain_name>>` where `<<cluster_name>>` and `<<domain_name>>` match the `cluster_name` and `domain_name` variables configured in the `group_vars/all/vars.yml` file.

Backup and restore

`etcd` is the key-value store for OpenShift Container Platform, which persists the state of all resource objects. It is highly recommended that you back up your cluster's `etcd` data regularly and store in a secure location, ideally outside the OpenShift Container Platform environment. Do not take an `etcd` backup before the first certificate rotation completes, which occurs 24 hours after installation, otherwise the backup will contain expired certificates. After the initial certificate rotation, next rotation occurs every 15 days or so thereafter and the certificates that are being rotated have an expiration of approximately 30 days. So it is recommended that you backup `etcd` every 2.5 weeks.

It is also recommended to take `etcd` backups during non-peak usage hours, as it is a blocking action.

Backup configuration and playbook

The playbook `backup_etcd.yml` is used to back up `etcd` assets. The playbook can also be used to back up your hosts and variable files, as well as the files generated by the initial deployment in the installation directory `install_dir`. If you need to re-deploy your cluster from scratch, it is very convenient to have a backup of these files.

Recovering from lost master hosts

It is possible to use the `etcd` backup to recover from the scenario where one or more master nodes have been lost. This includes situations where a majority of master hosts have been lost, leading to `etcd` quorum loss and the cluster going offline. The supported restore procedure assumes that you have at least one healthy master host.

The procedure is documented in the section Recovering from lost master hosts in the Deployment Guide and is based on the official documentation at https://docs.openshift.com/container-platform/4.1/backup_and_restore/disaster_recovery/scenario-1-infra-recovery.html. You need to check the latest version of the OpenShift documentation for any updates to this procedure.

Note

The procedure can be used to validate that your `etcd` backup has succeeded, but it is highly invasive in that it requires you to destroy master nodes and would render your cluster unusable for the duration of the procedure.

Summary

This document has described how to architect and deploy Red Hat OpenShift Container Platform on HPE SimpliVity, using Ansible playbooks to quickly install and deploy a production-ready container environment. This deployment provisions a highly available Kubernetes cluster with monitoring, logging and backup services and persistent data support. Red Hat OpenShift Container Platform offers automated operations, upgrades, and life-cycle management for every part of your container stack including the CoreOS operating system, Kubernetes, and cluster services and applications. The result is a more secure, always-up-to-date Kubernetes application platform, without the headaches of manual and serial upgrades, or downtime. Deploying this solution will help you increase developer productivity and innovation, while reducing operational complexities through built-in automation.

Appendix A: Bill of Materials

The following BOM contains electronic license to use (E-LTU) parts. Electronic software license delivery is now available in most countries. Hewlett Packard Enterprise recommends purchasing electronic products over physical products (when available) for faster delivery and for the convenience of not tracking and managing confidential paper licenses. For more information, please contact your reseller or a Hewlett Packard Enterprise representative.

Note

Part numbers are at time of publication and subject to change. The bill of materials does not include complete support options or other rack and power requirements. If you have questions regarding ordering, please consult with your Hewlett Packard Enterprise Reseller or Hewlett Packard Enterprise Sales Representative for more details. hpe.com/us/en/services/consulting.html.

Table 8 shows a representative BOM for HPE SimpliVity node on a single HPE ProLiant DL380 Gen10 Server, so you will need to multiply the quantities by the number of nodes you intend to deploy. This solution uses three nodes, which is the minimum recommended by Red Hat and Hewlett Packard Enterprise.

Table 8. Bill of Materials

Quantity	Part number	Description
1	Q8D81A	HPE SimpliVity 380 Gen10 Node
1	Q8D81A 001	HPE SimpliVity 380 Gen10 VMware Solution

Quantity	Part number	Description
1	826870-L21	HPE DL380 Gen10 Intel Xeon-Gold 6132 (2.6GHz/14-core/140W) FIO Processor Kit
1	826870-B21	HPE DL380 Gen10 Intel Xeon-Gold 6132 (2.6GHz/14-core/140W) Processor Kit
2	Q8D84A	HPE SimpliVity 192G 6 DIMM FIO Kit
1	Q5V86A	HPE SimpliVity 380 for 6000 Series Small Storage Kit
1	P01366-B21	HPE 96W Smart Storage Battery (up to 20 Devices) with 145mm Cable Kit
1	804331-B21	HPE Smart Array P408i-a SR Gen10 (8 Internal Lanes/2GB Cache) 12G SAS Modular Controller
1	700751-B21	HPE FlexFabric 10Gb 2-port 534FLR-SFP+ Adapter
1	BD505A	HPE iLO Advanced 1-server License with 3yr Support on iLO Licensed Features
1	Q8A60A	HPE OmniStack 2P Small SW
1	720865-B21	HPE 2U Cable Management Arm for Ball Bearing Rail Kit
1	867809-B21	HPE Gen10 2U Bezel Kit
1	826703-B21	HPE DL380 Gen10 SFF Systems Insight Display Kit
1	720863-B21	HPE 2U Small Form Factor Ball Bearing Rail Kit
1	H1K92A3 R2M	HPE iLO Advanced Non Blade Support

Resources and additional links

HPE Reference Architectures, hpe.com/info/ra

HPE SimpliVity, hpe.com/info/simplivity

Deployment Guide for Red Hat OpenShift Container Platform on HPE SimpliVity, https://hewlettpackard.github.io/OpenShift-on-SimpliVity

Ansible playbooks for Red Hat OpenShift Container Platform on HPE SimpliVity, https://github.com/HewlettPackard/OpenShift-on-SimpliVity

HPE Servers, hpe.com/servers

HPE Storage, hpe.com/storage

HPE Networking, hpe.com/networking

HPE Technology Consulting Services, hpe.com/us/en/services/consulting.html

To help us improve our documents, please provide feedback at hpe.com/contact/feedback.

a50000461enw, October 2019