Reference Configuration for Red Hat OpenShift Container platform on HPE Synergy Composable Infrastructure

Automated deployment of bare-metal servers using HPE Synergy Image Streamer and Ansible Tower by Red Hat

Complete approval table before submitting paper to Uno for QA. Table to be removed by Uno before publishing

RA ID	Reviewer's Name	BU/Title	Approval required Y/N	Date Sent	Date Approved
00000	Jason Shiplett	Engineering Product Owner	Y	00/00/0000	00/00/0000
	Ka Wai Leung	Product Mgr	Y		
	Ana Porras	Synergy Product Manager	Y		
	<Red Hat approvers>				

Technical white paper

Contents

Executive summary

Organizations are looking for a complete solution to deploy and manage containers at scale, efficiently using resources while providing ease of scalability and high availability along with improving visibility and control of their infrastructure. Container technology is rapidly changing the way organizations develop, deploy, and manage applications. Red Hat® OpenShift Container Platform provides organizations with a reliable platform for deploying and scaling container based applications. This paper provides architectural guidance for deploying, scaling and managing a Red Hat OpenShift environment on Hewlett Packard Enterprise Synergy Composable Infrastructure. It describes:

- How to leverage HPE Synergy strengths in rapid provisioning along with the automation capabilities of Red Hat Ansible Tower to provide a one-click deployment of a Red Hat OpenShift Container Platform on HPE Synergy.
- How to deploy an application and monitor the status of the Red Hat OpenShift environment with CloudForms including ChargeBack reporting.
- How to scale the infrastructure to allow for deployment of additional services.

Target audience: This document is intended for systems engineers, systems administrators, architects, and installers responsible for installing and maintaining Red Hat OpenShift Container Platform on a large scale running on Hewlett Packard Enterprise Synergy Composable Infrastructure. The reader of this document should be familiar with Red Hat OpenShift Container Platform, Red Hat Satellite, Ansible Tower, Red Hat Enterprise Linux, HPE OneView, and HPE Synergy Composable Infrastructure.

Introduction

This Reference Configuration describes a Red Hat OpenShift Container platform deployment on HPE Synergy Composable Infrastructure and includes details on how the environment was configured and links to download the Red Hat Ansible workflows and HPE Image Streamer artifacts used to automate the deployment. The document describes best practices and detailed instructions on configuring the environment for a successful deployment.

In this solution, Red Hat OpenShift Container platform is automatically deployed on physical servers. The management software used to deploy this solution, including Red Hat Ansible Tower, Red Hat Satellite, and Red Hat CloudForms (used to manage the environment), can be deployed either on physical servers or virtual machines.

Solution overview

This Reference Configuration provides a solution architected for Red Hat OpenShift Container Platform on bare metal HPE Composable Infrastructure and integrations to provide automated server provisioning and container management as shown in Figure 1. This creates a scalable and highly available platform for Red Hat OpenShift components.

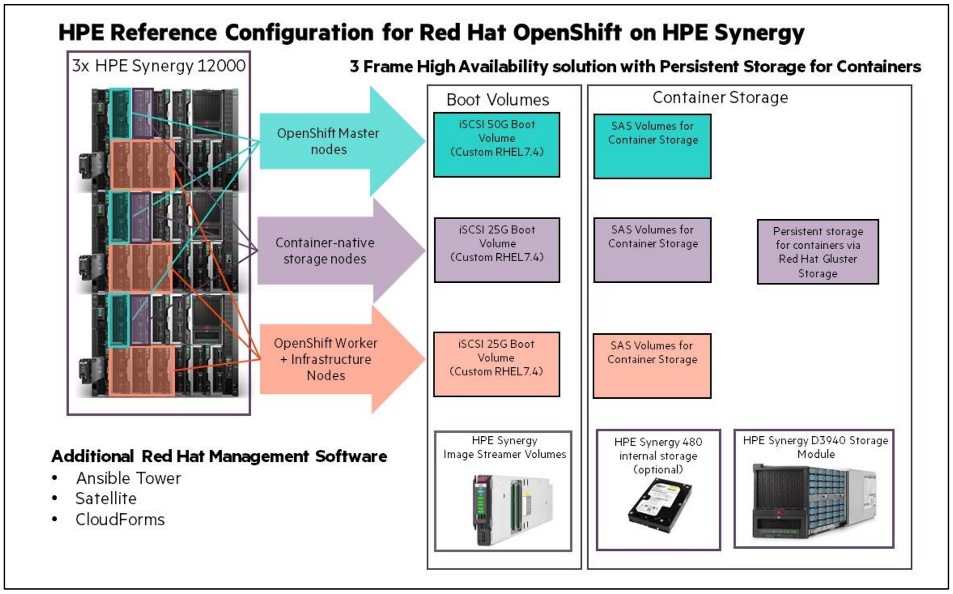


Figure 1: Red Hat OpenShift high availability deployment on HPE Synergy

This solution uses Red Hat Ansible Tower and Red Hat Satellite in combination with HPE OneView and HPE Image Streamer to automate the deployment of the following on physical servers:

- 3 OpenShift Master nodes (manage the Kubernetes cluster and schedule pods to run on worker nodes.)
- 3 OpenShift Container-native storage nodes (manage the glusterfs storage)
- 3 OpenShift Worker nodes (run the application pods)
- 3 OpenShift Infrastructure nodes (
- 1 OpenShift Load Balancer/NFS node (runs HAProxy to distribute requests, hosts image registry)
- 2 Additional OpenShift Worker nodes for scaling

Golden images and automated deployment plans configured and stored in HPE Synergy Image Streamer enable the rapid deployment of bare metal OpenShift infrastructure. The bare metal servers boot from HPE Image Streamer volumes and make use of (optional) HPE Synergy 480 internal storage and the HPE Synergy D3940 for ephemeral container storage. The HPE Synergy D3940 is used for persistent storage via Red Hat Gluster Storage.

Red Hat Ansible Tower workflows deploy playbooks utilizing the Ansible Modules for HPE OneView to apply server profiles to available HPE Synergy compute modules. The server profiles identify the server hardware type, networking, storage and OS deployment plan to apply to the server. After the OS has been deployed and customized via HPE Image Streamer, additional playbooks pull the required components from the Satellite server, deploy Red Hat OpenShift, and configure Red Hat Gluster storage.

Red Hat CloudForms Management Engine was used to manage the OpenShift container platform deployment. CloudForms helps define and maintain performance for the OpenShift deployment through policies, quotas and system profiles. After enabling cluster metrics on OpenShift deployments, CloudForms can collect node, pod and container metrics. Workloads and capacity can be tracked through SmartState analytics, chargeback and reports.

A sample microservice application based on JBoss Enterprise Application Platform 7 was downloaded and deployed onto Red Hat OpenShift Container Platform. The application was then monitored via Red Hat CloudForms.

Solution components

The solution hardware and software components used to build the HPE Reference Configuration for Red Hat OpenShift on HPE Composable Infrastructure are detailed in this section.

Solution hardware

The following hardware components were utilized in this Reference Configuration as listed in Table 1.

Table 1: Hardware components

Component	Purpose
HPE Synergy 12000 Frame	Infrastructure for compute, storage, fabric and management
HPE Synergy Composer	Infrastructure management
HPE Synergy Image Streamer	Infrastructure deployment
HPE Synergy 480 Gen10 Compute nodes	OpenShift Master and worker nodes
HPE Synergy D3940 Storage Module	Composable storage for OpenShift
HPE FlexFabric 5900AF Switch	Top of Rack network connectivity

HPE Synergy

HPE Synergy, the first platform built from the ground up for Composable Infrastructure, empowers IT to create and deliver new value instantly and continuously. This single infrastructure reduces operational complexity for traditional workloads and increases operational velocity for the new breed of applications and services. Through a single interface, HPE Synergy composes compute, storage and fabric pools into any configuration for any application. It also enables a broad range of applications from bare metal to virtual machines to containers, and operational models like hybrid cloud and DevOps. HPE Synergy enables IT to rapidly react to new business demands.

HPE Synergy Frames contain a management appliance called the HPE Synergy Composer which hosts HPE OneView. HPE Synergy Composer manages the composable infrastructure and delivers:

- Fluid pools of resources, where a single infrastructure of compute, storage and fabric boots up ready for workloads and demonstrates self-assimilating capacity.
- Software-defined intelligence, with a single interface that precisely composes logical infrastructures at near-instant speeds; and demonstrates template-driven, frictionless operations.
- Unified API access, which enables simple line-of-code programming of every infrastructure element; easily automates IT operational processes; and effortlessly automates applications through infrastructure deployment.

HPE Synergy Composer

HPE Synergy Frame's unique design physically embeds HPE Synergy Composer powered by HPE OneView to compose compute, storage, and fabric resources in any configuration. HPE Synergy Composer provides the enterprise-level management to compose and deploy system resources to your application needs. This management appliance uses software-defined intelligence to aggregate compute, storage, and fabric resources in a manner that scales to your application needs, instead of being restricted to the fixed ratios of traditional resource offerings. HPE Synergy template-based provisioning enables fast time to service with a single point for defining compute module state, pooled storage, network connectivity, and boot image.

HPE OneView is a comprehensive unifying platform designed from the ground up for converged infrastructure management. A unifying platform increases the productivity of every member of the internal IT team across servers, storage, and networking. By streamlining processes, incorporating best practices, and creating a new holistic way to work, HPE OneView provides organizations with a more efficient way to work. It is designed for open integration with existing tools and processes to extend these efficiencies.

HPE OneView is instrumental for the deployment and management of HPE servers and enclosure networking. It collapses infrastructure management tools into a single resource-oriented architecture that provides direct access to all logical and physical resources of the solution. Logical resources include server profiles and server profile templates, enclosures and enclosure groups, and logical interconnects and logical interconnect groups. Physical resources include server hardware blades, networking interconnects, and computing resources.

The HPE OneView converged infrastructure platform offers a uniform way for administrators to interact with resources by providing a RESTful API foundation. The RESTful APIs enable administrators to utilize a growing ecosystem of integrations to further expand the advantages of the integrated resource model that removes the need for the administrator to enter and maintain the same configuration data more than once and keep all versions up to date. It encapsulates and abstracts many underlying tools behind the integrated resource model, so the administrator can operate with new levels of simplicity, speed, and agility to provision, monitor, and maintain the solution.

The Ansible Modules for HPE OneView provide an interface to manage all HPE OneView resources and were used in the development of this solution.

HPE Synergy Image Streamer

HPE Synergy Image Streamer is a new approach to deployment and updates for composable infrastructure. This management appliance works with HPE Synergy Composer for fast software-defined control over physical compute modules with operating system and application provisioning. HPE Synergy Image Streamer enables true stateless computing combined with the capability for image lifecycle management. This management appliance rapidly deploys and updates infrastructure.

HPE Synergy Image Streamer adds a powerful dimension to 'infrastructure as code'—the ability to manage physical servers like virtual machines. In traditional environments, deploying an OS and applications or hypervisor is time consuming because it requires building or copying the software image onto individual servers, possibly requiring multiple reboot cycles. In HPE Synergy, the tight integration of HPE Synergy Image Streamer with HPE Synergy Composer enhances server profiles with images and personalities for true stateless operation.

HPE Synergy Composer, powered by HPE OneView, captures the physical state of the server in the server profile. HPE Synergy Image Streamer enhances this server profile (and its desired configuration) by capturing your golden image as the 'deployed software state' in the form of bootable image volumes. These enhanced server profiles and bootable OS plus application images are software structures ('infrastructure as code')—no compute module hardware is required for these operations. The bootable images are stored on redundant HPE Synergy Image Streamer appliances, and they are available for fast implementation onto multiple compute nodes at any time. This enables bare metal compute modules to boot directly into a running OS with applications and multiple compute nodes to be quickly updated.

Figure 2 shows an overview of the process to use Red Hat Ansible Tower workflows to select HPE OneView server profile templates, identify available compute hosts and automatically deploy Red Hat OpenShift ready servers. The Server Profile specifies the required networking, storage and firmware as well as the OS deployment plan from HPE Synergy Image Streamer. After the customized images are deployed, Ansible playbooks are used to install and configure the entire Red Hat OpenShift Container platform.

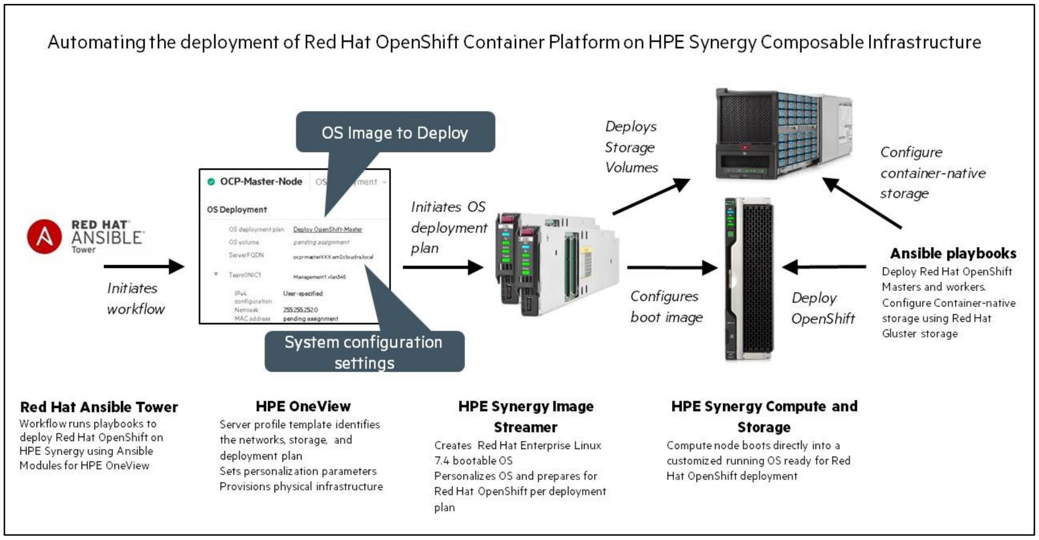


Figure 2: Deploying Red Hat OpenShift Container platform with HPE OneView, HPE Synergy Image Streamer and Red Hat Ansible Tower

HPE Synergy D3940 Storage Module

Each HPE Synergy D3940 Storage Module utilizes HPE Smart Array technology to accelerate performance, plus RAID protection and encryption to improve security and availability providing a fluid pool of storage resources for the composable infrastructure. Additional capacity for compute modules is easily provisioned and intelligently managed with integrated data services for availability and protection. The 40 SFF drive bays per storage module can be populated with 12 G SAS or 6 G SATA drives.

The HPE Synergy D3940 Storage Module provides local storage to compute resources and can meet the demands of a wide range of data workloads. Pooled storage resources provide the flexibility and performance needed to accommodate a wide range of workloads. Changes such as updating firmware are automatically implemented with the infrastructure online significantly reducing errors and delivering real-time compliance.

For this Reference Configuration the HPE Synergy D3940 Storage Module was used both for container images, as well as for Container-native storage using Red Hat Gluster storage. The server profiles for OpenShift Masters, Workers, Infrastructure, Load Balancer, and Container-Native storage nodes include the configuration of local storage. Either the SY480 internal storage or the D3940 based storage can be used for container ephemeral storage. Configuring the storage in this manner minimizes the use of limited storage on the HPE Synergy Image Streamer. The server profiles for OpenShift Container-Native storage nodes also include the configuration of persistent storage volumes using Red Hat Gluster storage.

HPE Synergy 480 Gen10 Compute Module

The HPE Synergy 480 Gen10 Compute Module delivers superior capacity, efficiency, and flexibility in a two-socket, half-height form factor to support demanding workloads. Powered by Intel® Xeon® Scalable Family of processors, up to 3TB DDR4, more storage capacity and controllers and a variety of GPU options, the HPE Synergy 480 Gen10 Compute Module provides a composable compute resource that can be self-discovered, quickly provisioned, easily managed, and seamlessly redeployed to deliver the right compute capacity for changing workload needs.

HPE Synergy 480 Gen10 Compute Module provides flexible enterprise solutions that allow IT to quickly and confidently implement changes through intelligent template-based operations, reducing downtime and errors from manual operations. The HPE Synergy Gen10 Compute Module provides unmatched threat protection with Hardware Root of Trust and unique chain of trust architecture to protect, detect and recover firmware, and Intelligent System Tuning to smooth performance and match customer workloads through security features that include:

- iLO5 with Silicon Root of Trust for a secure start
- Tamper-proof updates with digital validation
- Enhanced scripting and API capabilities
- Secure firmware recovery
- Standards compliance
- Supply-chain attack detection

Solution management software

The following software components were utilized in this Reference Configuration as listed in Table 2 and Table 3.

Table 2: Red Hat components

Component	Version
Red Hat Ansible Tower	3.2.1
Red Hat CloudForms	5.8.1.5
Red Hat Enterprise Linux	7.4
Red Hat Gluster Storage	3.3
Red Hat OpenShift	3.6
Red Hat Satellite	6.2

Table 3. Hewlett Packard Enterprise solution management software

Component	Version
HPE Synergy Composer	3.10.07
HPE Synergy Image Streamer	3.10.02
HPE Synergy Image Streamer artifacts for Red Hat OpenShift	HPE-RHEL-7.4-OpenShift-11-.xx-2017.zip

Ansible Tower by Red Hat

Ansible is a powerful configuration management and orchestration tool, which can be used to automate complicated infrastructure or application deployment tasks. At its core, ansible ships with hundreds of modules to manage different host types and software. These hosts can be traditional servers running Red Hat Enterprise Linux and other operating systems, as well as an impressive count of network switches and storage arrays. Modules are also included to interact with APIs (application program interface) directly, or wrapped up in a separate module to present a standard set of arguments. For example, the openstack (*os_*) modules are provided to orchestrate an OpenStack cloud.

An ansible **playbook** is built by defining tasks. A given task will call a module with the desired set of arguments. The playbook is then run against an inventory which specifies the hosts to run those tasks.

Ansible Tower takes those ansible constructs, and presents them in a clean, modern web GUI. This Reference Configuration used Ansible Tower by Red Hat to orchestrate the provisioning of HPE Synergy 480 Gen10 servers, integration with HPE OneView and HPE Image Streamer for installation of Red Hat Enterprise Linux 7.4, and deployment of Red Hat OpenShift Container Platform with Container-native storage. This was accomplished using Ansible Tower inventory, templates, roles, tasks, and workflows, to execute the multiple ansible playbooks required for this Reference Configuration. The installation of Ansible Tower is beyond the scope of this Reference Configuration. Refer to the Ansible Tower by Red Hat documentation for complete instructions on the installation and configuration of Ansible Tower by Red Hat.

Red Hat CloudForms

Red Hat CloudForms delivers the insight, control and automation that enterprises need to address the challenges of managing virtual and containerized environments. This technology enables enterprises to build and operate new environments as well as improve visibility and control of existing environments.

For this Reference Configuration, Red Hat CloudForms was used to monitor the Red Hat OpenShift Container Platform through integration with the Hawkular services running on Red Hat OpenShift. In this Reference Configuration the Hawkular monitoring and logging services are enabled as part of the initial Red Hat OpenShift installation via ansible playbooks. The installation of Red Hat CloudForms is beyond the scope of this Reference Configuration. Refer to the Red Hat CloudForms deployment planning guide for more information.

Details on how Red Hat CloudForms was configured to monitor and manage the Red Hat OpenShift infrastructure are found in Appendix E: Using Red Hat CloudForms to monitor the Red Hat OpenShift Container Platform.

Red Hat Gluster Storage

Containerized Red Hat Gluster storage provides the persistent storage for Red Hat OpenShift containers in this Reference Configuration. In this hyper-converged solution, the storage containers that host Red Hat Gluster Storage co-reside with the compute containers and serve out storage from the hosts that have local or direct attached storage to the compute containers. This solution integrates Red Hat Gluster Storage deployment and management with OpenShift services. As a result, persistent storage is delivered within an OpenShift pod that provides both compute and file storage.

Container-Native Storage for OpenShift Container Platform is built around three key technologies:

- OpenShift provides the platform as a service (PaaS) infrastructure based on Kubernetes container management. Basic OpenShift architecture is built around multiple master systems where each system contains a set of nodes.
- Red Hat Gluster Storage provides the containerized distributed storage based on Red Hat Gluster Storage 3.3 container. Each Red Hat Gluster Storage volume is composed of a collection of bricks, where each brick is the combination of a node and an export directory.
- Heketi provides the Red Hat Gluster Storage volume life cycle management. It creates the Red Hat Gluster Storage volumes dynamically and supports multiple Red Hat Gluster Storage clusters.

Refer to Container-Native Storage for OpenShift Container Platform for more details on Red Hat Gluster Storage.

Red Hat OpenShift

Red Hat OpenShift Container Platform is a platform as a service (PaaS) offering from Red Hat that brings together Docker and Kubernetes, and provides an API to manage these services. Red Hat OpenShift Container Platform helps application development and IT operations teams modernize applications, deliver new services, and accelerate development processes.

OpenShift Container Platform provides developers with an optimal platform for provisioning, building, and deploying applications and their components in a self-service fashion. With automated workflows like the source-to-image (S2I) process, it is easy to get source code from version control systems into ready-to-run, docker-formatted container images. OpenShift Container Platform integrates with continuous integration (CI) and continuous delivery (CD) tools, making it an ideal solution for any organization.

OpenShift Container Platform gives IT operations a secure, enterprise-grade Kubernetes that provides policy-based control and automation for applications. Cluster services, scheduling, and orchestration provide load-balancing and auto-scaling capabilities. Security features prevent tenants from compromising other applications or the underlying host. And because OpenShift can attach persistent storage directly to Linux® containers, IT organizations can run both stateful and stateless applications on one platform.

For more information on Red Hat OpenShift Container Platform, see the Red Hat OpenShift Container Platform 3.6 datasheet.

Red Hat Satellite

Red Hat Satellite is used to manage Red Hat infrastructure. For this Reference Configuration, Red Hat Satellite provides a local mirror of all available Red Hat packages for faster software delivery inside the data center. A built in DNS server provides automated DNS entry when a host is created, and deletion when the host is decommissioned.

The configuration of Red Hat Satellite for the Red Hat OpenShift deployment is described in Appendix D: Configuring Red Hat Satellite.

Best practices and configuration guidance for the solution

Red Hat OpenShift deployment best practices utilizing HPE Synergy Image Streamer and Ansible Tower

This Reference Configuration outlines how to configure a deployment of Red Hat OpenShift on physical servers as shown in Figure 3. This configuration provides an optimized solution with highly available OpenShift Master, OpenShift Container Native Storage and OpenShift Worker deployments.

This solution is ideal for organizations who are developing cloud native applications or refactoring existing applications using containers and want to maximize resource, performance and cost efficiencies. Additional Red Hat OpenShift master or worker nodes can be rapidly deployed as needed on the fluid pool of HPE Synergy resources.

HPE Synergy configuration

The three HPE Synergy frames used for this Reference Configuration include HA deployment of HPE Synergy Composer and HPE Synergy Image Streamer as shown in Figure 3. The HPE Synergy D3940 Storage modules are used for ephemeral storage for container deployments and for the Container-native storage.

The initial deployment of the solution deploys Synergy compute modules deployed as:

- 13x Synergy 480 Gen10 servers, decomposed in:
  - Red Hat OpenShift Masters: 3x servers, 1 per frame
  - Red Hat Container Native Storage: 3x servers, 1 per frame
  - Infrastructure nodes: 3x servers, 1 per frame
  - Red Hat OpenShift workers 3x servers, 1 per frame
  - Load Balancer: 1x server
- 2x Synergy 480 Gen 10 servers available in the environment were used to demonstrate scaling out Red Hat OpenShift workers.

Each HPE Synergy D3940 was configured with 16x 300GB SAS HDD allowing for all container and Red Hat Gluster volumes to be created using redundant volumes.

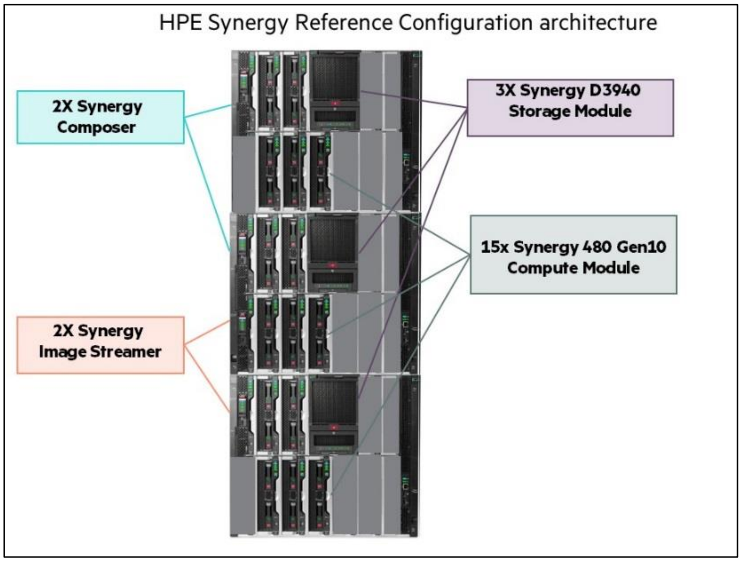


Figure 3: HPE Synergy configuration

HPE Synergy Image Streamer best practices

Best practices for HPE Synergy Image Streamer deployment and usage include:

- As shown in Figure 3, redundant HPE Synergy Image Streamers were deployed as part of this Reference Configuration, and are required for a production installation. If you need to scale your environment beyond 3 HPE Synergy Frames, be sure to follow HPE recommended guidelines to ensure your configuration meets minimum requirements.
- Deploy container images to a storage system such as the Synergy D3940 used for this Reference Configuration. Do not deploy the container images to your root filesystem on the HPE Synergy Image Streamer to avoid large and rapidly-growing smart clones.
- Create artifact bundles of your working deployment plans, build plans, build scripts and golden images and download them to alternate storage so they can be restored if necessary.
- As existing images and deployment plans are updated, be sure to remove any that are no longer used to preserve available storage. Monitor the available storage from the Deployment Appliances page under Storage.

Utilizing HPE Synergy Image Streamer and Ansible Tower to deploy bare metal servers for Red Hat OpenShift

For this Reference Configuration, HPE Image Streamer is used to deploy the initial images and perform customizations in preparation for Red Hat OpenShift deployment. Ansible Tower is used to deploy and configure OpenShift on the provisioned servers and storage volumes.

The deployment of Red Hat OpenShift is greatly simplified through the use of HPE Synergy Image Streamer and Ansible Tower. The process to use Ansible Tower and HPE Synergy Image Streamer to automatically provision bare metal servers requires the following steps:

1. Configure the networking in HPE OneView
2. Download and import the HPE Image Streamer artifacts for Red Hat OpenShift
3. Configure Red Hat Satellite
4. Create golden images for OpenShift Masters, Container Native Storage workers and OpenShift Workers
5. Customize the HPE supplied deployment plans
6. Create Server Profile Templates in HPE OneView
7. Configure Ansible Tower for Red Hat OpenShift Container Platform deployment
8. Deploy Red Hat OpenShift Container platform from Ansible Tower
9. Monitor the solution from Red Hat CloudForms

Additional optional steps include:

1. Configure Container-native storage
2. Lifecycle management for OpenShift Container Platforms on HPE Synergy
3. Deploying a sample application

Configure the networking in HPE OneView

Two networks were configured for the Red Hat OpenShift deployment. A corresponding network set was created which includes a single network as a best practice. The network set is used in the server profiles. Using network sets inside the server profile allows for modification of the networks included in the network set without having to modify a server profile. This makes it easier to add or remove networks without having to power off the server.

The following Ethernet networks are required in the Synergy environment, for usage in the server profiles for OpenShift deployment:

- Public management network (ns_mgmt)
- Image Streamer Deployment network (ns_deployment)

Table 4 summarizes the configuration for each network. Networks were created with a preferred bandwidth of 2.5 Gb/second and maximum bandwidth of 20 Gb/second because they share a single Virtual Connect SE 40 Gb F8 module.

Table 4: Networks for Docker deployment

Network set name	VLAN	Subnet ID	Uplink set	Description
ns_mgmt	60	10.60.0.0/16	sus_prod	Management network for Red Hat OpenShift deployment
ns_deployment	10	10.10.10.0/24	i3s-iscsi	HPE Synergy Image Streamer deployment network

Download and import the HPE Image Streamer artifacts for Red Hat OpenShift
Hewlett Packard Enterprise has created HPE Synergy Image Streamer artifacts and published them to HPE's GitHub location for customer download. The following artifact bundle was used in the development of this Reference Configuration., and HPE-RHEL-7.4-OpenShift-2017-11-27-v3.1.zip. Import the downloaded artifact bundle into HPE Synergy Image Streamer and extract the contents. The artifact bundles include:

Commented [CC1]: Update with hyperlink when available

- Deployment plans
- OS build plans
- Plan scripts

Configure Red Hat Satellite
This solution requires an installation of Red Hat Satellite and licenses to Red Hat OpenShift software. Pre-requisite packages are pulled from Red Hat Satellite for the golden image creation. The HPE Image Streamer scripts register deployed servers with Red Hat Satellite and the Ansible playbooks pull down the required OpenShift packages as part of server deployment. See Appendix D: Configuring Red Hat Satellite for details on how to configure Red Hat Satellite.

Create golden images for OpenShift Masters, Container Native Storage workers and OpenShift Workers
The golden images for Red Hat OpenShift deployment are the captured content of a previously deployed RHEL 7.4 boot disk customized with the pre-requisites for OpenShift deployment. Two golden images were captured. OpenShift Masters require 50GB boot disks. OpenShift Container Native Storage and Workers require 25GB. The only difference between the two golden images is the initial volume size created before OS Deployment. The process to capture an image automatically removes system-specific information such as hostname and IP addresses, enabling the image to be re-deployed and automatically customized for deployment using HPE Synergy Image Streamer artifacts such as plan scripts, OS build plans and deployment plans.

The process to create the RHEL 7.4 golden images is described in Appendix B: Creating golden images for Red Hat OpenShift deployment.

Customize the HPE supplied deployment plans
Four deployment plans are provided for Red Hat OpenShift deployments.

- HPE-RHEL7.4-OpenShiftMaster
- HPE-RHEL7.4-OpenShiftWorker
- HPE-RHEL7.4-OpenShiftCNS
- HPE-RHEL7.4-OpenShiftLBNFS

The deployment plans for Red Hat OpenShift as shown in Figure 4 needs to be customized before it can be used. Make copies of the plans and edit the values for the Plan Attributes to match the settings for your environment. You will want to hide some of the custom attributes and their values so that they are pre-configured when server profile templates are created. For example, if you always specify the same NTP server, set this value and hide the corresponding custom attributes and values. Details on the custom attribute definitions and values are found in Appendix C: Setting the custom attributes in the OpenShift deployment plans. Set the golden image which corresponds to the OpenShift component for the selected deployment plan. The 50GB golden image is used for the OpenShift Master and the 25GB golden image is used for Worker, Load Balancer, and Container-native storage nodes.

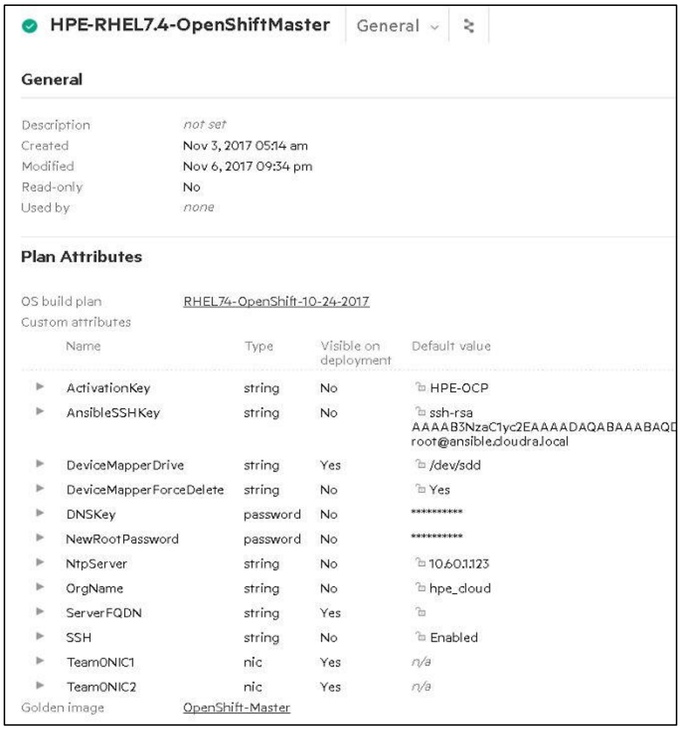


Figure 4: Deployment plan for OpenShift Master

Create Server Profile Templates in HPE OneView

From HPE OneView, create server profile templates for the OpenShift Master, OpenShift Container Native Storage, Load Balancer and OpenShift worker nodes. By using the Server Profile Template feature within HPE OneView you can specify and maintain a single configuration for the system firmware, BIOS, and boot-order at time of initial deployment as well as orchestrate updates to that configuration as needed. This provides a location to centrally manage and update configuration settings, such as system firmware, and provides assurance that each server is running with the same configuration and has event and health data being exposed up to HPE OneView. The server profile requirements for Red Hat OpenShift deployment are listed in Table 5.

Table 5: Red Hat OpenShift server profile requirements

Component	Image Streamer Boot Volume	Storage	RAID level	Networks

Red Hat OpenShift Master	50GB	300GB container	1	Mgmt
Red Hat OpenShift Worker / Infrastructure	25GB	300GB container	1	Mgmt
Red Hat OpenShift Load Balancer / NFS	25GB	300GB nfs	1	Mgmt
Red Hat Container-native storage	25GB	300GB container 300GB gluster	1 6	Mgmt

Note:
The 300GB RAID 6 storage configuration for Gluster provides 600GB of useable storage

The server profile template for OpenShift Master is shown in Figure 5.

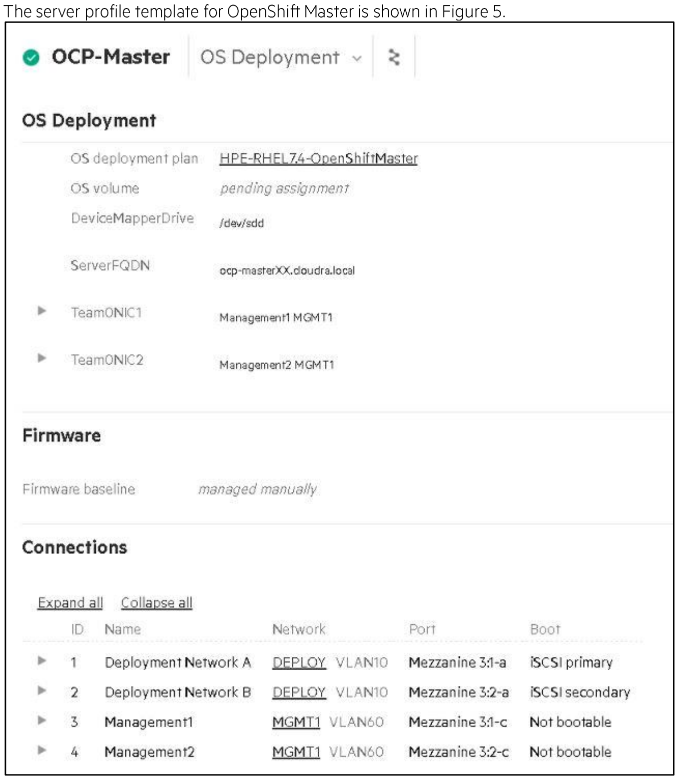


Figure 5: Server profile for OpenShift master

- Specify the OS deployment plan you created for the OpenShift Master node. The Deployment Network is automatically added when an OS Deployment plan is selected.

- Under Connections, add 2 instances of the management network for NIC teaming. Specify either the management network or the management network set.
- Under Local Storage, select to edit SAS Mezz 1 storage controller. Select the checkboxes for Manage Mezz 1 storage controller and Re-initialize controller on next profile application and click Select Create logical drive.
- Creation of logical drives depends on whether or not you wish to use internal server storage or external storage from the D3940. Select the storage you wish to use for container storage as shown in Figure 6. For this solution RAID 1 was used for container storage.

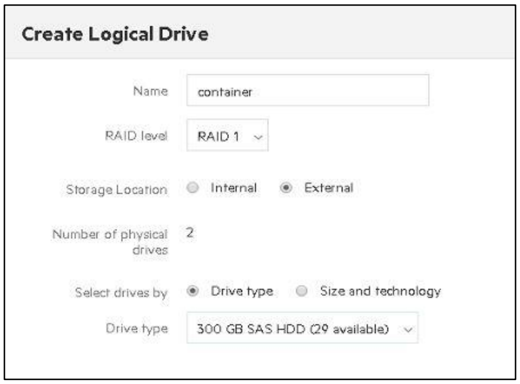


Figure 6: Create ephemeral storage for containers

- After the connections and storage are configured, preset values for the custom attributes. Set a value for FQDN, Team0NIC1 and Team0NIC2. The values for NIC attributes will be overwritten by the Ansible playbooks so it does not really matter what values you set here. It is more important to set Custom Attributes correctly in the Deployment Plan.

Follow the same process to create Server Profile templates for OpenShift Workers, OpenShift Load Balancer and OpenShift Container Native Storage. Each Server Profile template must include the corresponding Deployment plan for the OpenShift component. Each requires the same management network configuration and a logical drive for container storage. Container-native storage nodes must be configured with an additional logical drive for Gluster. When creating the logical drive for Gluster, set the RAID level to RAID 6. The storage for Gluster must be configured to use an external logical driver (D3940) as shown in Figure 7.

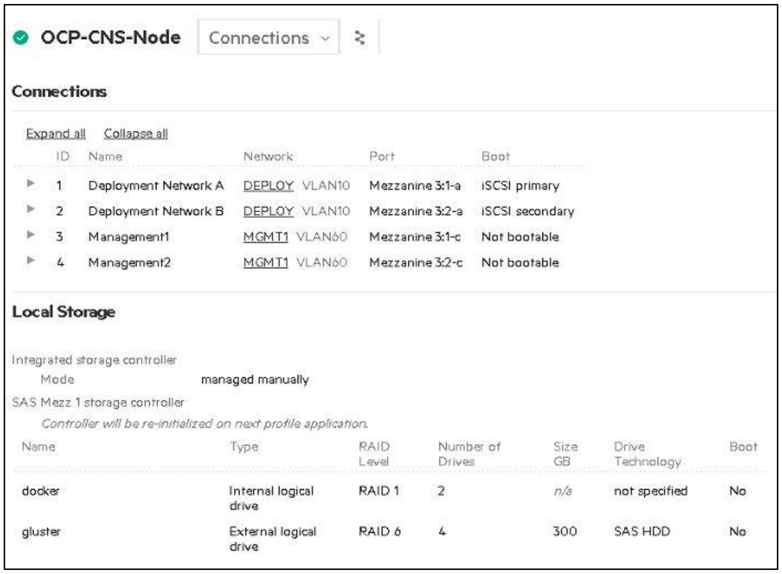


Figure 7: Connections and Storage information from Server Profile Template for OpenShift Container-native storage

After the server profile templates have been created, perform test deployments manually.

1. Create a server profile from the OCP-Master profile template you created and apply it to an available compute module.
2. Power on the server
3. Use ssh to connect to the server.
4. Use the `df` and `cat proc partitions` commands to identify which partition is used for boot and which is used for docker containers. Be sure the custom attribute DeviceMapperDrive is set to the partition you want to use for container deployments and not to the boot partition. The device specified is wiped to ensure a successful installation.
5. Use nslookup to verify that the hostname you provided in the server profile template resolves correctly
6. Verify that the server registered with Satellite. If it did not register automatically, validate that the AcitvationKey and OrgName custom attributes are set correctly in your Deployment plan.
7. Power off the test deployment, delete the server profile and remove the entry from Satellite to free up the license.
8. Follow the same process to deploy an OCP-CNS node using the server profile template you created.
9. The OCP-CNS node has both container and gluster storage. Validate that the values you set for DeviceMapperDrive match what you see from logging in via ssh. The DeviceMapperDrive parameter requires that the first value is the partition for container storage and the second value is for gluster storage on the D3940. It should be specified as a comma separated value, for example: `/dev/sdc`, `/dev/sdb`
10. Power off the server, delete the server profile and remove the entry from Satellite to free up the license.

Configure Ansible Tower for Red Hat OpenShift Container Platform deployment

This section provides information about the Ansible playbooks created for this Reference Configuration and how the Ansible Tower Workflow ties it all together. The set of playbooks were created with modularity in mind. Rather than create one monolithic playbook to interact with the various services, a role was created for each major step. This makes debugging sections easier, as well as allowing for re-use of certain roles in other infrastructures. For instance, the provisioning playbook could be used in another Ansible Tower workflow, with the OpenShift Container deployment swapped out for another infrastructure stack.

The Red Hat OpenShift Container Platform installer is based on Ansible, and an advanced install exposes this to a systems administrator. Inventory and variables can be configured and the playbook can be run using the standard ansible-playbook command. The use of these playbooks is well documented on the OpenShift site. This Reference Configuration only covers integrating and running those playbooks in Ansible Tower. Playbooks to handle all of the provisioning steps leading up to the OpenShift installation are available in a GitHub repository. Some modifications will be required to work in different environments, outlined below.

NOTE:
The scripts described in this reference architecture and provided in the associated GitHub repository are not supported by Red Hat. They merely provide a mechanism to build out your infrastructure.

Configuring Ansible Tower for Red Hat OpenShift deployment

This section provides the installation and configuration details for installing OpenShift Container Platform using Ansible Tower.

Clone and Modify GitHub Repository

In order to deploy the playbooks created for this reference architecture, some modifications of the content are required. Notably, a password file encrypted with `ansible-vault` needs to be copied to `/var/lib/awx/projects/ocp-on-synergy/roles/passwords/vars/passwords.yml`. Environment specific changes should be made to the variables found under `./group_vars/`.

If an organization has an existing git infrastructure (such as gitlab) they may choose to clone the public repository and maintain their changes internally. Otherwise, the repository can be forked on github to track the requisite changes. In either case, the modified playbooks must be stored in a git repository that is accessible to Ansible Tower.

HPE OneView Modules

The HPE OneView Ansible modules are not a core module yet, so they are not included in a standard installation. They must be installed on the Ansible Tower server along with the required python packages.

First, install the HPE OneView SDK via pip:

```
curl -o get-pip.py https://bootstrap.pypa.io/get-pip.py
python get-pip.py
pip install hpOneView dnspython
```

Then clone the modules and make them accessible to Ansible Tower:

```
git clone https://github.com/HewlettPackard/oneview-ansible.git /tmp/oneview-ansible
mkdir -p /usr/share/ansible/oneview-ansible
rsync -avh --progress /tmp/oneview-ansible/library /usr/share/ansible/oneview-ansible
```

Modified nsupdate module

The set of playbooks written for this Reference Configuration require a modified nsupdate module to support one to many DNS creation. A pull request has been submitted to support this feature and is merged in the Ansible 2.4 release. The updated file is included in the ocp-on-synergy GitHub repository for those using Ansible 2.3 or earlier and included during playbook execution via a custom ansible.cfg file also stored in the repository.

Credentials

Ansible typically communicates to a target host via SSH, using a public key. Ansible Tower can store and encrypt these credentials to run playbooks. An SSH key pair is generated:

```
ssh-keygen -f tower
Generating public/private rsa key pair.
```

```
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in tower.
Your public key has been saved in tower.pub.
The key fingerprint is:
...
```

And then stored in Ansible Tower by clicking the gear icon, then credentials and finally +ADD as shown in Figure 8:

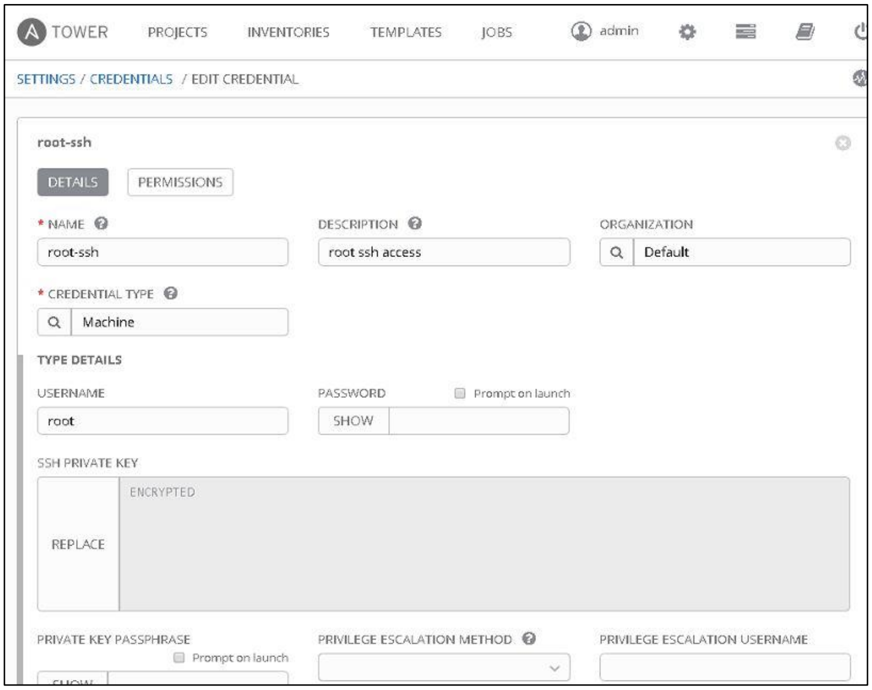


Figure 8: Configure Ansible ssh keys

The password used to encrypt the password playbook is also stored under 'Vault Password'. The public SSH key is deployed via Red Hat Satellite when a host is provisioned. In some cases, the git repository may require login details, which can also be stored safely in Ansible Tower.

Projects

The locally maintained git repository ocp-on-synergy is added to Ansible Tower as shown in Figure 9.

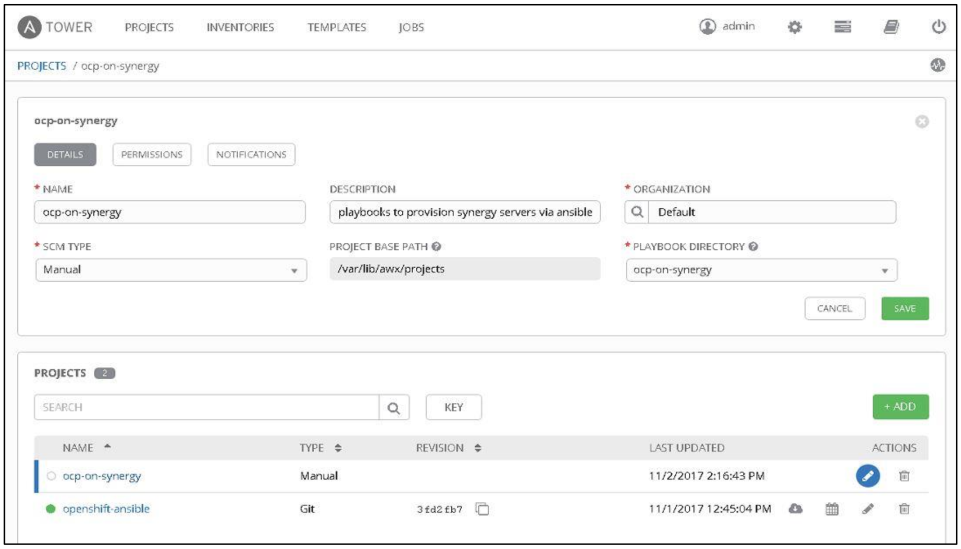


Figure 9: ocp-on-synergy project

As well as the publicly accessible openshift-ansible playbook as shown in Error! Reference source not found..

Figure 10: openshift-ansible project

The list of hosts and variables for this reference architecture can be entered via the web GUI, however for complex playbooks with a lot of variables, this can be cumbersome. Ansible Tower provides a handy CLI tool that can be used to import this data. Before using the tool, the empty inventory must first be created in the web GUI as shown in Figure 11.

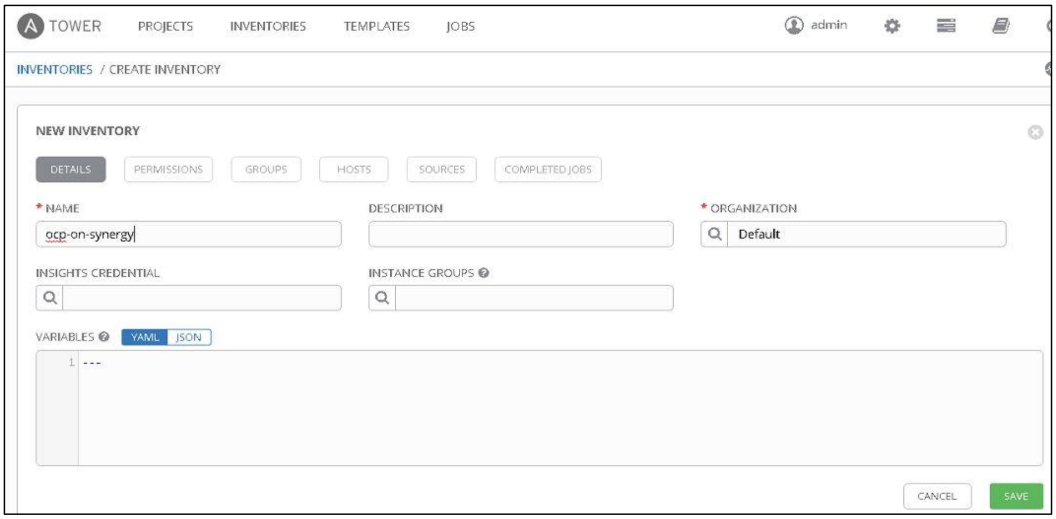


Figure 11: Create inventory

Creating the inventory from the browser interface creates the host file and variable files on the Ansible Tower server in `/etc/ansible/ocp-on-synergy`. Sample hosts file and variable files as well as instructions to import the files can be found in the GitHub repository or in Appendix G: Ansible playbooks and customization. Modify the sample files to meet your organization's requirements and import the variable files.

Templates

Once the credentials, projects and inventories are configured, the job templates can be created in Ansible Tower. From Templates, select to add a job template as shown in Figure 12.

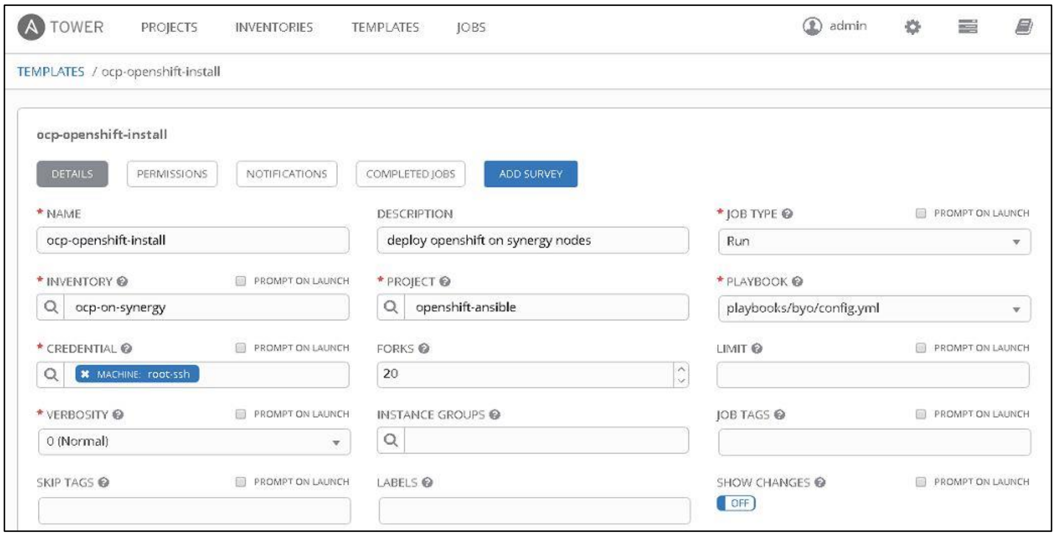


Figure 12: Creating templates in Ansible Tower

The following job templates were created as shown in Table 6. Create the additional job templates with the specifications shown below.

Table 6: Ansible job templates

Template name	Project	Playbook	Description	Forks
ocp-add_to_oneview	ocp-on-synergy	playbooks/add_to_oneview.yaml	add a new node to OneView for scaling	default
ocp-cleanup	ocp-on-synergy	playbooks/clean.yaml	Clean up all deployments	20
ocp-fix_inventory	ocp-on-synergy	playbooks/fix_inventory.yaml	move new host in to normal 'nodes' host group	default
ocp-openshift-install	openshift-ansible	playbooks/byo/config.yml	deploy openshift on synergy nodes	20
ocp-openshift-scaleup	openshift-ansible	playbooks/byo/openshift-node/scaleup.yml	scale up openshift cluster with new node	default
ocp-postdeploy	ocp-on-synergy	playbooks/postdeploy.yaml	ocp postdeployment tasks	20
ocp-predeploy	ocp-on-synergy	playbooks/predeploy.yaml	ocp predeployment tasks	20
ocp-provisioning	ocp-on-synergy	playbooks/provisioning.yaml	provisioning nodes via OneView for ocp	20
ocp-provisioning-limited	ocp-on-synergy	playbooks/provisioning.yaml	limit provisioning to new_nodes group	20
ocp-scaledown	ocp-on-synergy	playbooks/scaledown.yaml	remove node from openshift then from OneView	default

The ocp-scaledown job template requires a survey to supply the required variables. Use the Add Survey button to configure the job to prompt for a target_node and supply the information as shown in Table 7:

Table 7: Survey prompts for ocp-down

Prompt	Description	Answer Variable Name	Answer Type	Required
Target Hostname	Enter a hostname to remove from the cluster	target_node	String	Checked

Once the job templates are created, you can create workflow templates. Two workflow templates were created:

1. ocp-on-synergy: End to End provisioning of OpenShift on Synergy hardware

Use the workflow editor to add the following steps as shown in Figure 13.

   a. ocp-cleanup
   b. ocp-provisioning
   c. ocp-predeploy
   d. ocp-openshift-install
   e. ocp-postdeploy

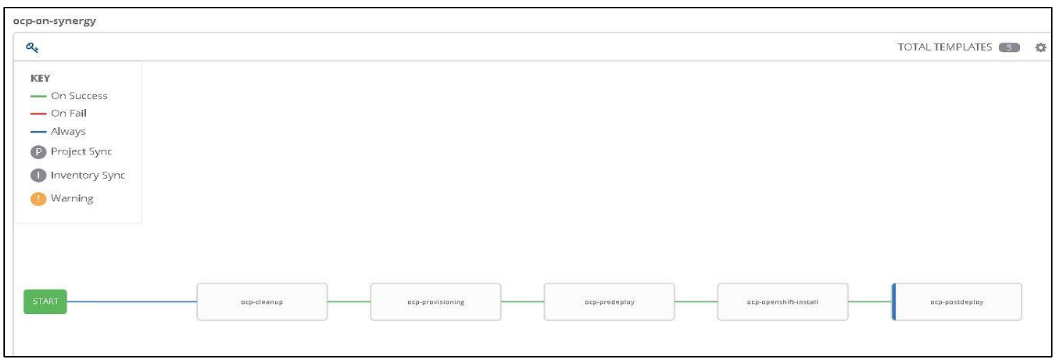


Figure 13: Ansible workflow to deploy Red Hat OpenShift

2. ocp-scaleup: Scale up the OCP cluster

   This workflow expects to be supplied with a host name and IP for the new node. The variables are supplied using the Add Survey button. Configure the prompts as shown in Table 8:

Table 8: Survey prompts for ocp-scaleup

Prompt	Description	Answer Variable Name	Answer Type	Required
New Hostname	Enter a hostname for the new node	new_node	String	Checked
New IP	Enter the IP address for the new node	new_ip	String	Checked

Use the workflow editor to add the following steps as shown in Figure 14.

   a. ocp-add_to_onview
   b. ocp-provisioning-limited (provisioning applied to new_nodes group)
   c. ocp-openshift-scaleup

d. ocp-fix_inventory

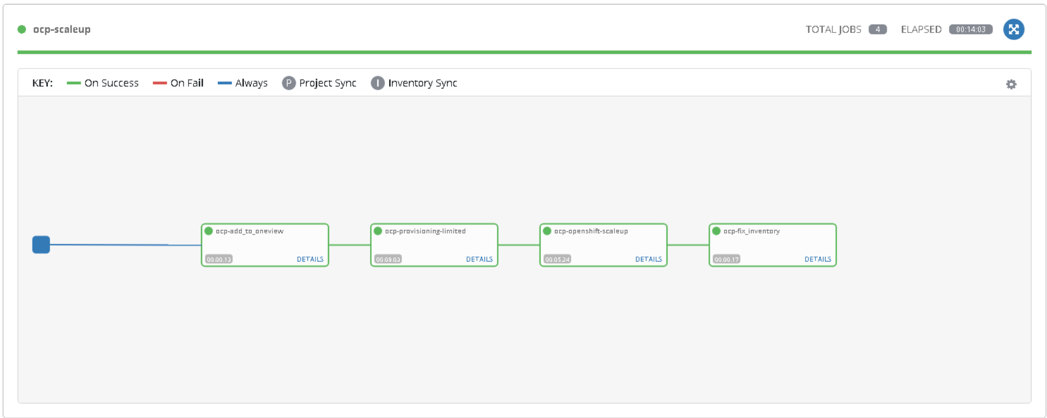


Figure 14: Ansible workflow to scale up Red Hat OpenShift deployment

Deploy Red Hat OpenShift Container platform from Ansible Tower

Once the workflows and playbooks are configured in Ansible Tower, deployment of the Red Hat OpenShift Container platform can be performed with a single click. From Ansible Tower, select the Templates tab and select the ocp-on-synergy workflow Launch icon to start the deployment as shown in Figure 15.

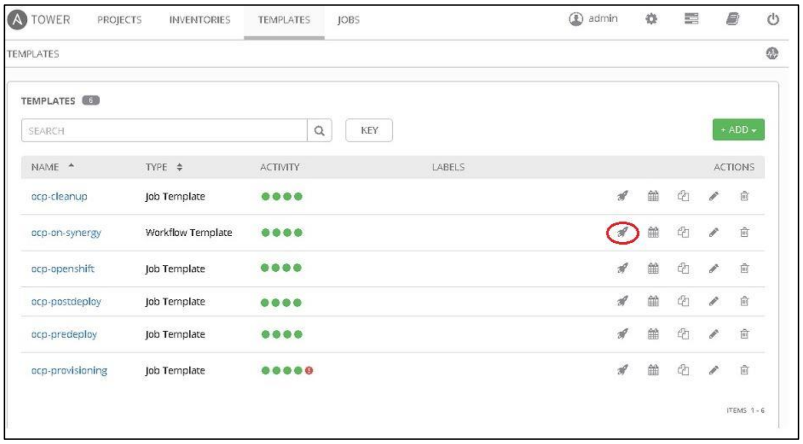


Figure 15: Deploying Red Hat OpenShift from Ansible Tower

Track the status of the workflow from the Jobs tab. The output of the ocp-provisioning step is shown in Figure 16.

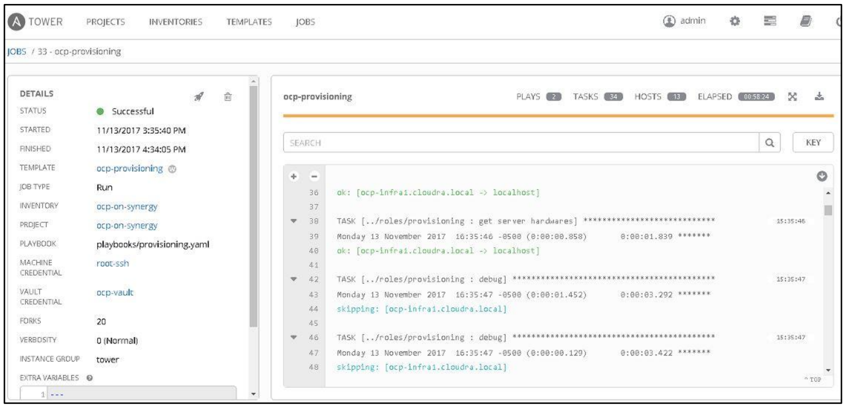


Figure 16: Viewing deployment status in Ansible Tower

Once the job completes successfully, you will be able to deploy container services in your environment either via the Red Hat OpenShift web interface or via the command line interface from one of the master nodes. With the deployment complete the Red Hat OpenShift Container Platform will resemble the deployment illustrated in the image below. The deployment will have three master nodes running the etcd cluster, three infrastructure nodes, and three Container-native storage nodes, three worker nodes and a Loadbalancer NFS server node as shown in Figure 17. These nodes provide the basic services and infrastructure for the Red Hat OpenShift Container Platform cluster. Additional application nodes can be added as needed to scale out the cluster. The registry for container applications is running on two of the infrastructure nodes. Load Balancer NFS node is running HA Proxy to provide load balancing of external traffic among the master nodes. This node is also running an NFS server to provide persistent storage for the registry. The routing services are running on two of the infrastructure nodes as well. The routers provide routing services between the external network and the container network. There are three nodes running Container-native storage to provide containers with persistent storage.

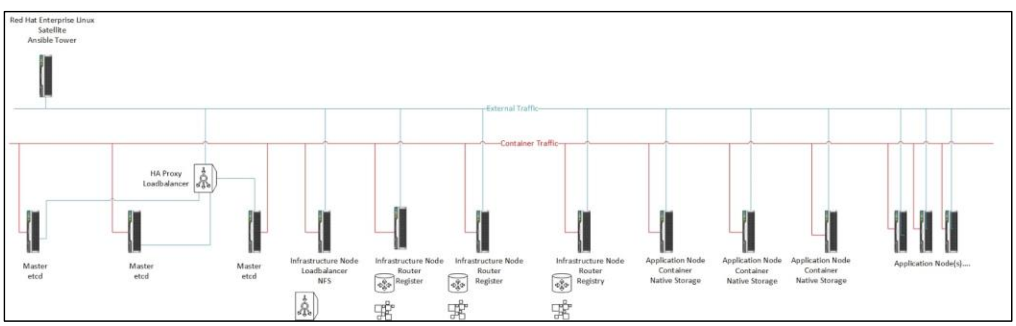


Figure 17: Fully deployed Red Hat OpenShift Container Platform

After deployment the OpenShift cluster server nodes can be viewed and managed through the HPE OneView management inteface as shown below in Figure 18.

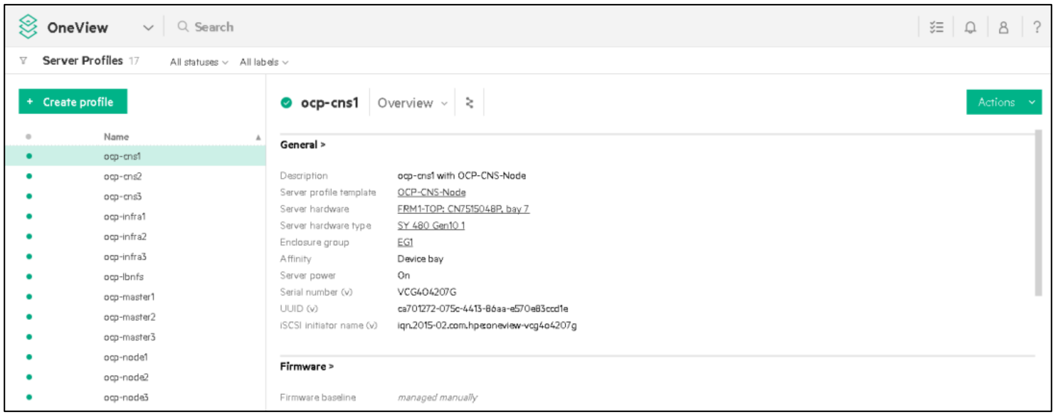


Figure 18: HPE OneVew Server Profiles

Validation and troubleshooting procedures are found in

Appendix H: Red Hat OpenShift Container Platform deployment validation and troubleshooting.

Monitor the solution from Red Hat CloudForms

Red Hat CloudForms was used to monitor the deployment of Red Hat OpenShift on HPE Synergy. Details on configuring Red Hat OpenShift as a provider, creating custom reports and monitoring newly deployed applications is found in Appendix E: Using Red Hat CloudForms to monitor the Red Hat OpenShift Container Platform.

Configure Container-native storage

The Red Hat OpenShift Container Platform cluster was deployed with a three node containerized glusterfs cluster. A default storage class named glusterfs-storage was created to support persistent storage for containers. This can be viewed by executing `oc describe storageclass` as shown below.

```
oc describe storageclass
Name:            glusterfs-storage
IsDefaultClass:  Yes
Annotations:     storageclass.kubernetes.io/is-default-class=true
Provisioner:     kubernetes.io/glusterfs
Parameters:      resturl=http://heketi-storage-
glusterfs.paas.cloudra.local,restuser=admin,secretName=heketi-storage-admin-
secret,secretNamespace=glusterfs
Events:          <none>
```

The `oc create` command will be used to create persistent volume claims on the glusterfs-storage storage class.
Create a text file that will be used to the *oc create* command to create a persistent volume claim using the glusterfs-storage class. In this example the text file was named pvc.yaml and the contents of the text file are displayed below.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 name: glusterpvc
spec:
```

```
 accessModes:
  - ReadWriteOnce
 storageClassName: glusterfs-storage
 resources:
   requests:
     storage: 5Gi
```

Use the pvc.yaml file with the oc create command to create the persistent volume claim.

```
oc create -f pvc.yaml
persistentvolumeclaim "glusterpvc" created
```

Use the oc get pvc command to view the persistent volume claim.

```
oc get pvc glusterpvc
NAME         STATUS    VOLUME                                     CAPACITY   ACCESSMODES
STORAGECLASS        AGE
glusterpvc   Bound     pvc-8482cfed-cec9-11e7-ab79-2a77ef000417   5Gi        RWO           glusterfs-
storage   6m
```

Use the *oc get pv* command to view persistent volume that was dynamically created on the glusterfs-storage class.

```
oc get pv pvc-8482cfed-cec9-11e7-ab79-2a77ef000417
NAME                                       CAPACITY   ACCESSMODES   RECLAIMPOLICY   STATUS   CLAIM                STORAGECLASS        REASON   AGE
pvc-8482cfed-cec9-11e7-ab79-2a77ef000417   5Gi        RWO           Delete          Bound    default/glusterpvc   glusterfs-storage            8m
```

Lifecycle management for OpenShift Container Platforms on HPE Synergy

If demand for container deployments exceeds the capacity of the existing worker nodes, new worker nodes can be added using the ocp-scaleup workflow template.

- From Ansible Tower, select the Templates tab and select the Launch button for the ocp-scaleup workflow template.
- Provide a new hostname and an IP address and then select Launch as shown in Figure 19.

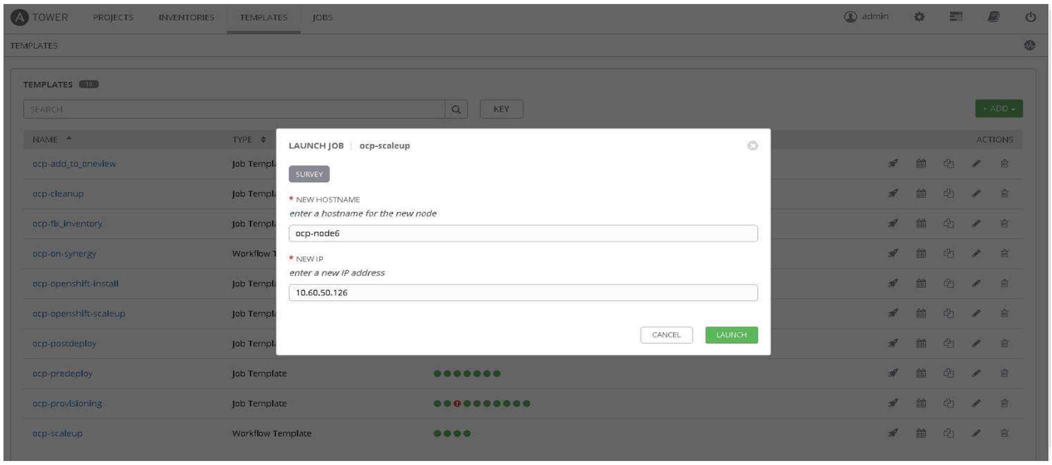


Figure 19: Scale up worker nodes with Ansible Tower

Follow the progress of the scale-up workflow from the Jobs tab. Once the process completes, the new worker node is available for container deployments.

Similarly, if you no longer need the extra capacity and wish to free up compute resources, use the ocp-scaledown Job Template and supply the hostname for the server you wish to remove from the cluster.

Deploying a sample application

A sample Microservice application was deployed on the Red Hat OpenShift platform. Details on deploying the sample application are found in Appendix F: Deploying a sample application

Performance benefits of Red Hat OpenShift deployment using HPE Synergy Image Streamer

This Reference Configuration includes the artifacts that were used to automate the deployment of bare metal servers in preparation for Red Hat OpenShift deployment. The golden images used for deployment are configured with the pre-requisite packages installed and firewall configuration settings. The plan scripts automatically set the hostname and networking as well as configure the storage for container deployments. Thru the use of golden images and plan scripts, the overall deployment time is significantly reduced compared to a manual deployment or a scripted installation using kickstart files. The process to clone the volume takes less than 3 minutes. The worker OS deployment times shown below include the time to power on the server, POST, and boot the OS

Table 9: Additional Worker deployment comparison

Deployment method	Deployment type	Deployment time
HPE Synergy Image Streamer	Volume smart clone	5 minutes
Red Hat Satellite OS Kickstart deployment	Scripted OS install	15 minutes
ISO image deployment plus manual configuration	Manual install and configuration	120 minutes (estimated)

End to end deployment of the entire 13 node OpenShift cluster takes approximately 1 hour to provision the operating systems on the 13 nodes. An additional hour is required for OpenShift deployment and customization. This could be compared to a manual installation provided by a consulting or services organization. This type of deployment could take from 3 – 5 days. The automation also reduces the chances of operator or installer induced errors.

Summary

The world of IT is always evolving and system administrators are being called upon to deliver increasingly complex systems. The hardware at their disposal needs to be flexible and adapt as well. HPE's Synergy platform, coupled with Red Hat Ansible Tower, empowers IT to fully automate - from bare metal to software installation - the deployment of complex systems.

This Reference Configuration has described the solution HPE and Red Hat have jointly created to help customers take full advantage of today's composable platforms. This solution provides IT with a one-click deployment of OpenShift Container Platform on HPE Synergy using Ansible Tower.

This Reference Configuration describes solution testing performed in November 2017.

Appendix A: Bill of materials

The following BOMs contain electronic license to use (E-LTU) parts. Electronic software license delivery is now available in most countries. HPE 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 an HPE 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 HPE Reseller or HPE Sales Representative for more details. hpe.com/us/en/services/consulting.html

Table 10: Bill of materials

Quantity	Part number	Description
		Rack and network infrastructure
1	BW908A	HPE 42U 600x1200mm Enterprise Shock Rack
4	AF522A	HPE Intelligent 8.6kVA/L15-30P/NA/J PDU
1	HC790A	HPE Integration Center Routg Service FIO
1	BW932A	HPE 600mm Rack Stabilizer Kit
1	BW909A	HPE 42U 1200mm Side Panel Kit
1	JG505A	HPE 59xx CTO Switch Solution
2	JG510A	HPE 5900AF 48G 4XG 2QSFP+ Switch
4	JD096C	HPE X240 10G SFP+ SFP+ 1.2m DAC Cable
2	JC680A	HPE 58x0AF 650W AC Power Supply
2	JC682A	HPE 58x0AF Bck(pwr) Frt(prt) Fan Tray
		Synergy 12000 3 Frame components
3	797740-B21	HPE Synergy12000 CTO Frame 1xFLM 10x Fan
3	798096-B21	HPE Synergy 12000F 6x 2650W AC Ti FIO PS
2	804353-B21	HPE Synergy Composer
3	804942-B21	HPE Synergy Frame Link Module
1	804938-B21	HPE Synergy 12000 Frame Rack Rail Option
1	804943-B21	HPE Synergy 12000 Frame 4x Lift Handle
18	TK738A	HPE 2.0m 250V 16A C19-C20 Sgl IPD Jpr Crd
2	804937-B21	HPE Synergy Image Streamer
		Synergy 480 Gen 10 compute components
15	871942-B21	HPE SY 480 Gen10 CTO Premium Cmpt Mdl (64Gb Memory)

Quantity	Part number	Description
30	873388-B21	HPE Synergy 480 Gen10 6130 Kit
30	873388-L21	HPE Synergy 480 Gen10 6130 Kit
30	815100-B21	HPe 32GB QUAD PC4-2166P-R Kit
15	871573-B21	HPE Smart Array P416ie-m
15	875242-B21	HPE Smart Array P416ie-m SAS Cable Kit
15	777430-B21	HPE Synergy 3820C 10/20Gb CAN
15	875242-B21	HPE 96W Smart Stor Battery 260mm Cbl Kit

Quantity	Part number	Description
		Synergy Fabric Components
2	794502-B23	HPE VC SE 40Gb F8 Module
4	779218-B21	HPE Synergy 20Gb Interconnect Link Mod
6	755985-B21	HPE Synergy 12Gb SAS Connection Module
		Synergy Composable Storage Components
3	835386-B21	HPE Synergy D3940 CTO Storage Module
3	757323-B21	HPE Synergy D3940 IO Adapter
30	785067-B21	HPE 300GB 12G SAS 10K 2.5in SC ENT HDD
		Cables and Transceivers
8	804101-B21	HPE Synergy Interconnect Link 3m AOC
2	720199-B21	HPE BLc 40G QSFP+ QSFP+ 3m DAC Cable
8	720193-B21	HPE BLc QSFP+ to SFP+ Adapter
8	455883-B21	HPE BLc 10G SFP+ SR Transceiver
8	AJ837A	HPE 15m Multi-mode OM3 LC/LC FC Cable
9	861412-B21	HPE CAT6A 4ft Cbl
2	838327-B21	HPE Synergy Dual 10GBASE-T QSFP+ 30m RJ45 Transceiver
		Red Hat licenses
13		Ansible Tower (managed nodes)
12		Red Hat OpenShift (includes Red Hat Enterprise Linux)
1		Red Hat Satellite
1		Red Hat Enterprise Linux 7 (required for loadbalancer)
1		Red Hat CloudForms

Commented [CC2]: How do we include the required RH licenses in the BOM?

Commented [KB3R2]: Ken to research
Clarify for smart management

Commented [KB4R2]:

Appendix B: Creating golden images for Red Hat OpenShift deployment

This section provides details on how to capture the golden images required for Red Hat OpenShift deployment. Two images will be captured: a 50GB image for Red Hat OpenShift masters and 25GB image for workers. The process to create the two golden images is exactly the same except for the initial volume size setting. The HPE-RHEL7.4-OpenShift artifact bundle contains the OS Deployment Plan which allows you to create an initial 50GB volume and an OS Build plan necessary to capture a RHEL7.4 image.

Commented [CC5]: Update with the hyperlink when available

Note: It is possible to use the same 50GB golden image for both OpenShift masters and workers. Creating a smaller golden image for OpenShift workers saves limited volume space on the Image Streamer. If desired, a separate image for the Load Balancer can be created. The Load Balancer image does not require installation of the OpenShift or docker packages.

Install Red Hat Enterprise Linux 7.4

1. Create a server profile with the Image Streamer deployment network and the management network. The management network is needed to allow for installation of the prerequisite packages which will be pulled from the Red Hat Satellite server.
2. Use the OS Deployment plan HPE Create Empty Volume, set the volume size to 51200 MB and add the management network as shown in Figure 20.

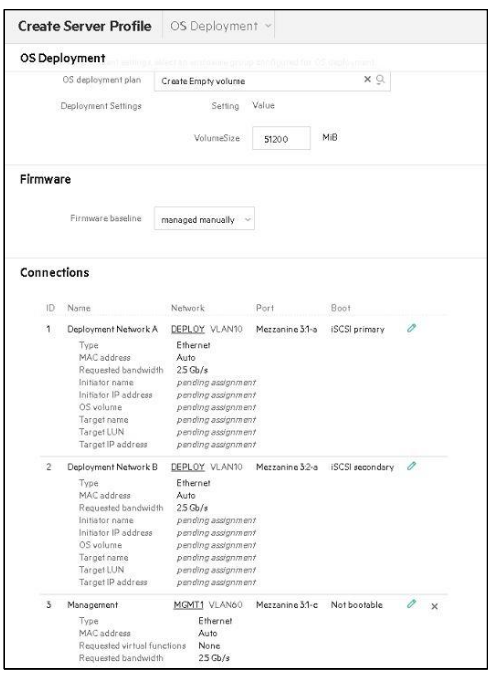


Figure 20: Creating the server profile for a golden image

3. Launch the iLO console for the server where the profile was assigned and use iLO virtual media to mount the RHEL7.4 iso image.

4. Power on the server and when the Red Hat Enterprise Linux installer screen appears, press the "e" key to edit the parameter which enables the RHEL 7.4 install kernel to recognize the Image Streamer iSCSI empty OS volume. Add the `rd.iscsi.ibft=1` kernel boot parameter as shown in Figure 21 and then press Ctrl x to start the installation.

Figure 21: Install Red Hat Enterprise Linux 7.4 from iso image

Set the desired defaults for keyboard and language. The installation source will be local media. Do not set a network or hostname in the golden image. Select Installation Destination and then from the Installation Destination window, select **I will configure the partitioning** as shown in Figure 22.

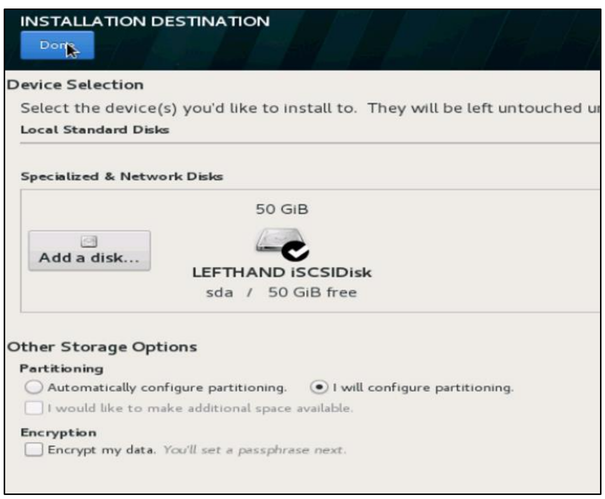


Figure 22: Selecting the iSCSI device for OS installation

5. Leave the default as LVM partitioning and then select to create the partitions automatically.
6. Change both the /boot and / partitions from File System type xfs to ext4 and select the Update Settings button to save the changes as shown in Figure 23.

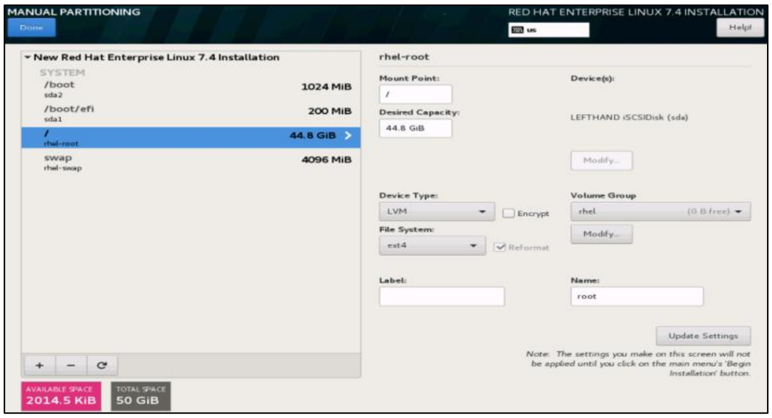


Figure 23: Set File System type for RHEL7.4

7. Set a default root password so you can log into the image after reboot and complete the configuration. When prompted, unmount the iso image and reboot the server.

Customize the OS deployment

Now that you have completed the basic OS deployment, the next steps customize the golden image including setting the required firewall settings and deploying the required packages. The steps in this section are documented in the Red Hat OpenShift Installation and Configuration guide but are listed here for ease of deployment.

1. Configure the management network manually so that you can install packages from the Satellite server. Once you have the network configured, connect to the server via ssh which will allow you to cut and paste commands.
2. Configure the firewall settings as follows:

```
firewall-cmd --add-port 4789/udp -permanent
firewall-cmd --add-port 8053/udp -permanent
firewall-cmd --add-port 8053/tcp --permanent
firewall-cmd --add-port 443/tcp --permanent
firewall-cmd --add-port 8443/tcp --permanent
firewall-cmd --add-port 10250/tcp --permanent
firewall-cmd --add-port 53/tcp --permanent
firewall-cmd --add-port 53/udp --permanent
firewall-cmd --reload
firewall-cmd --list-all   (to verify settings)
```

3. Edit /etc/resolv.conf to set the nameserver and search list. This will allow us to pull the required packages from the Satellite server. For example:

```
nameserver 10.10.173.45
```

```
search am2.cloudra.local
```

4. Install the katello package from the Satellite server, for example:

```
rpm -Uvh http://satellite.am2.cloudra.local/pub/katello-ca-consumer-latest.noarch.rpm
```

5. Register with Red Hat Satellite. The organization name and activation key can be retrieved from Red Hat Satellite from the Activation Keys page as shown in Figure 24, for example:

```
subscription-manager register --org="my-org" --activationkey="my-key"
```

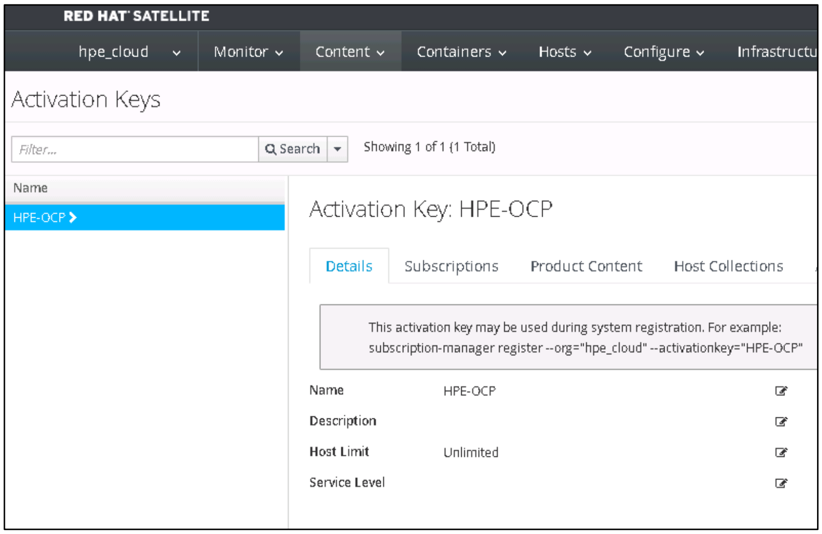


Figure 24: Red Hat Satellite Activation Keys

6. Set up the required packages as follows:

```
subscription-manager list --available --matches '*OpenShift*'
```

7. In the output from the previous command, find the pool ID for an OpenShift container Platform subscription and attach it:

```
subscription-manager attach --pool=<pool_id>
```

8. Disable all yum repositories:

```
subscription-manager repos --disable="*"

yum-config-manager --disable \*
```

9. Enable only the repositories required by OpenShift container Platform 3.6:3

```
subscription-manager repos \
--enable="rhel-7-server-rpms" \
--enable="rhel-7-server-extras-rpms" \
--enable="rhel-7-server-ose-3.6-rpms" \
--enable="rhel-7-fast-datapath-rpms"
```

10. Install required base packages:

```
yum install -y wget git net-tools bind-utils iptables-services bridge-utils bash-completion kexec-tools sos psacct

yum update

yum install -y atomic atomic-openshift-utils docker-1.12.6 gdisk
```

11. Edit the /etc/sysconfig/docker file and add --insecure-registry 172.30.0.0/16 to the OPTIONS parameter. For example:

```
OPTIONS='--selinux-enabled --insecure-registry 172.30.0.0/16'
```

12. The golden image must be unregistered from Satellite. When servers are deployed from the golden image, they will automatically be re-registered as part of the Image Streamer scripts.

```
subscription-manager unregister
```

13. The networking you configured on your golden image will be automatically cleaned out by the Capture Image process. Shut down the server gracefully.

Capture the deployed image

1. From HPE OneView, view the OS Deployment section of the server profile used to create the golden image and find the OS volume. This is the volume we will capture as a golden image
2. From the Image Streamer UI, select Golden Images and then Create golden image.
3. Provide a name for your golden image, such as OpenShift Master, select the OS volume you identified in Step 1 and select the Capture OS build plan RHEL-7.4-generalize and click Create as shown in Figure 25.

Figure 25: Create golden image

Creating the smaller golden image

Once the golden image has been created for OpenShift Master, you can modify the same server profile to start the process to create the 25GB volume for OpenShift workers and container-native storage nodes.

1. Edit the server profile and set the volume size to 20000MB.
2. A warning message will appear informing you that the existing volume will be deleted. Since you have already captured the golden image, click OK to proceed.

3. Follow the same procedures documented in this section to create the smaller golden image for OpenShift Workers.

After this process is complete you should have created two golden images, one 50GB OpenShift Master golden image and one 25GB OpenShift worker golden image.

Appendix C: Setting the custom attributes in the OpenShift deployment plans

The RHEL-7.4-OpenShift deployment plan custom attributes are the same for both Master and worker nodes. The values for all attributes are set the same for all deployment plans except for the DeviceMapperDrive attribute as described in Table 11.

Table 11: Custom Attributes for OpenShift deployment

Name	Description	Sample Value
ActivationKey	The Red Hat Satellite activation key. This name is used in combination with the OrgName to register with Red Hat Satellite. Figure 24 shows where this can be found on the Red Hat Satellite web interface. The activation key for the load balancer deployment plan does not need to include Red Hat OpenShift Container Platform. If you have a separate activation key that only includes Red Hat Enterprise Linux, you can use this key for the load balancer deployment plan.	HPE-OCP
AnsibleSSHKey	The ssh key from the Ansible server is copied to the authorized_keys file on deployed servers and allows Ansible scripts to run without login prompting. Key is found on the Ansible server at /root/.ssh/id_rsa.pub	
DeviceMapperDrive	The filesystem location for the container and Gluster volumes. This value varies depending on whether or not local storage is available. A single value should be specified for all but Container-native storage nodes. The first value is the location of the container (docker) storage. Use a comma to separate values. In the testing done for this reference configuration, Master, Infra, Load Balancer and worker nodes the value was /dev/sdb. For Container-Native storage nodes where the container storage was on internal volumes and the gluster storage was on external (D3940), the value was /dev/sdc,/dev/sdb.	/dev/sdb
DeviceMapperForceDelete	Forces a disk wipe of the drives specified in the DeviceMapperDrive attribute so logical volumes can be re-used. This attribute must be set to Yes or the installation will not work properly.	Yes
DNSKey	The DNS key from the DNS server which allows deployed servers to register with DNS. This information can be found in /etc/rndc.key file on the Satellite server. Specify the key name followed by the secret, separated by a colon	rndc_key: ikfhgqSdaPRLix34AVAMTg==
NewRootPassword	Password for the root user	
NtpServer	IP address for the NTP server. An NTP server is required or communication between containers could fail.	10.10.10.100
OrgName	The Red Hat Satellite organization name. This name is used in combination with the Activation Key to register with Red Hat Satellite. Figure 24 shows where this can be found on the Red Hat Satellite web interface.	
ServerFQDN	Fully qualified domain name. This value is set by the Ansible playbooks.	server1.acme.com
SSH	Enable the SSH service and root login over ssh. The default is Enabled.	Enabled
Team0NIC1	A static IP address for the deployed OpenShift server on your production network. Select Static User Assigned for this address. This value is set by the Ansible playbooks.	
Team0NIC2	This NIC is used for teaming. Leave the default value of static. No IP address is needed. This value is set by the Ansible playbooks.	

Appendix D: Configuring Red Hat Satellite

Red Hat Satellite was used in this Reference Configuration to provide access to the required repositories and to provide DNS services. The following sections provide an overview of the Satellite server configuration to support the deployment of Red Hat OpenShift:

Configuration to Support OpenShift Container Platform

Repositories

It is assumed that customers have an active Satellite configuration and access to the required subscriptions and repositories. In this reference architecture a manifest with the required Red Hat product was created in access.redhat.com. The manifest was then imported into Satellite by selecting Content > Red Hat Subscriptions > Manage Manifest > Choose File > Upload.

The required repositories are then enabled by selecting Content > Red Hat Repositories > RPMs and selecting the required repositories.

Next the selected repositories must be synchronized with Red Hat. This is done by selecting Content > Products > Select the product repository and choose Sync Now. Ensure that the Satellite server is synchronized to the required Red Hat products.

The following repositories are required to be available from the Red Hat Satellite server to deploy OpenShift with Container-native storage:

- Red Hat Gluster Storage 3.3 Server (RPMs)
- Red Hat Enterprise Linux Fast Datapath (RHEL 7Server) (RPMs)
- Red Hat Enterprise Linux 7 Server - Extras (RPMs)
- Red Hat OpenShift Container Platform 3.6 (RPMs)
- Red Hat Enterprise Linux 7 Server (RPMs)
- Red Hat Satellite Tools 6.2 (for RHEL 7 Server) (RPMs)

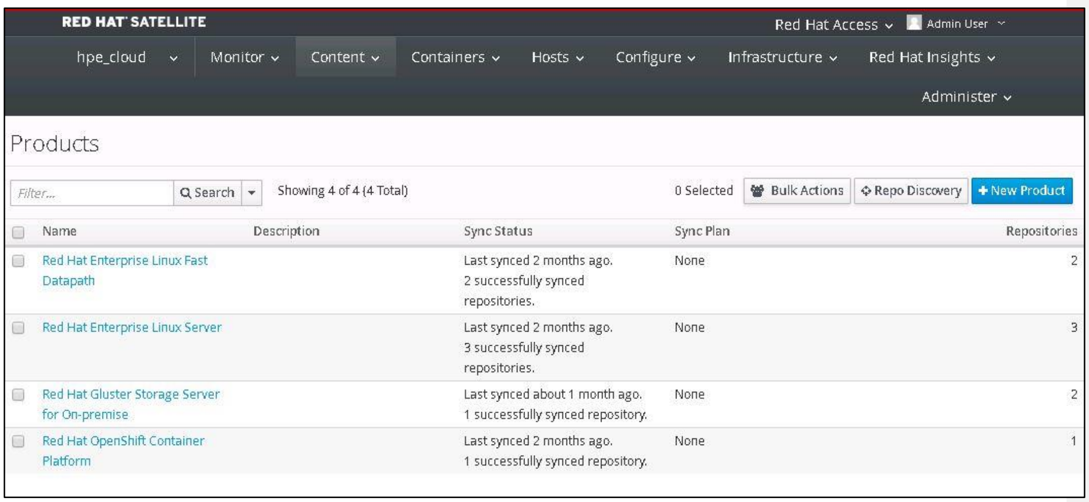


Figure 26: Required Red Hat Satellite Repositories

Next create an Activation Key by selecting Content > Activation Key > New Activation Key. Create the new Activation Key and ensure the required repositories are enabled as shown in **Figure 27:** Activation KeyFigure 27. The additional activation key shown in Figure 27 as HPE-RHEL is used for the loadbalancer and does not require the OpenShift subscriptions and product content.

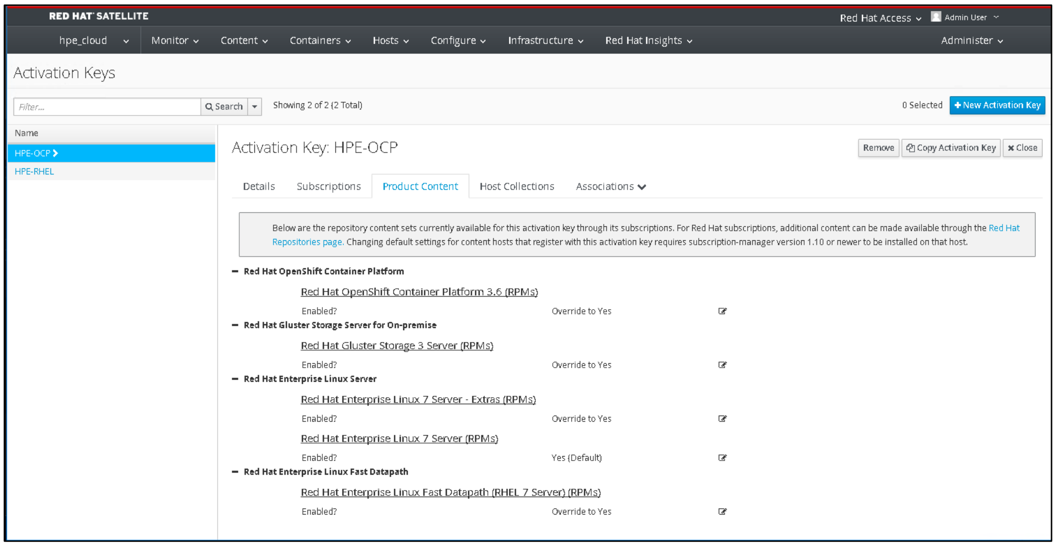


Figure 27: Activation Key

Appendix E: Using Red Hat CloudForms to monitor the Red Hat OpenShift Container Platform

This section provides details on how the OpenShift provider was configured in CloudForms and examples on creating reports to view the usage metrics collected. Refer to Integration with OpenShift Container Platform for full instructions on creating the service account and enabling metrics in Red Hat OpenShift Container Platform. A summary of steps is listed below.

In this Reference Configuration Red Hat CloudForms provides monitoring of the Red Hat OpenShift Container Platform through integration with the Hawkular services running on Red Hat OpenShift. In this Reference Configuration the Hawkular monitoring and logging services are automatically enabled as part of the initial OpenShift installation via ansible.

CloudForms requires a service account to be created on the Red Hat OpenShift Container Platform. The ansible postdeploy workflow `cloudforms.yaml` task in this Reference Configuration will create a service account named `cfadmin` and assign the OpenShift cluster-reader role to the cfadmin service account. A service token associated with the cfadmin service account must be generated. The service token will be used authenticate the CloudForms OpenShift provider to the Red Hat OpenShift Container Platform cluster. This service token will be specified in the CloudForms Container Provider form.

- Use ssh to connect to ocp-master1. Log in as system:admin and generate a service token using the service account. The service token will be used to provide access from CloudForms to the OpenShift cluster.

```
oc login -u system:admin

oc sa get-token -n openshift-infra cfadmin
```

Once the service token is available you can log in to CloudForms enable metrics collection, and add the OpenShift Container Provider information.

Log in to CloudForms as admin user. In the top right corner, click on the drop down menu under Administrator and then select Configuration. Change the settings under Server Roles to On as shown in Figure 28 for

- Capacity & Utilization Coordinator
- Capacity & Utilization Data Collector
- Capacity & Utilization Data Processor

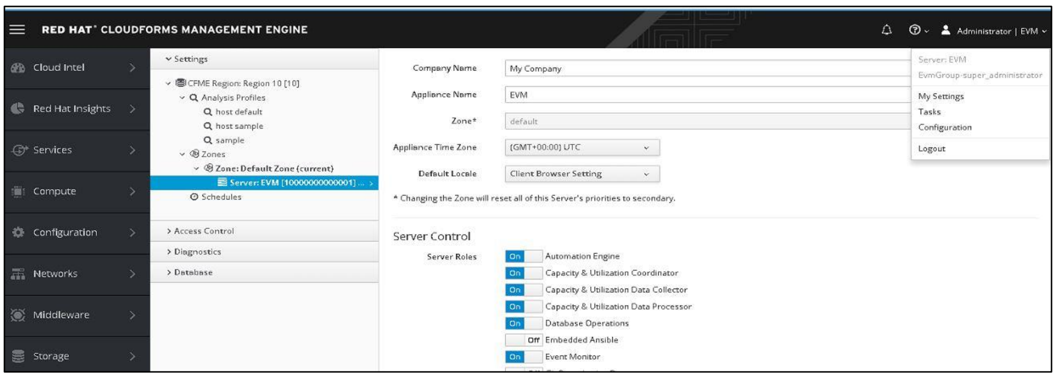


Figure 28: Enable Metrics collection

Save the changes and select Compute > Containers > Providers > Configuration > Add Existing Container Provider. Complete the form as shown in Figure 29.

- Specify the FQDN for the cluster as the Hostname.
- Cut and paste the token returned from the `oc sa get-token` command in step 4 above into the Token field.

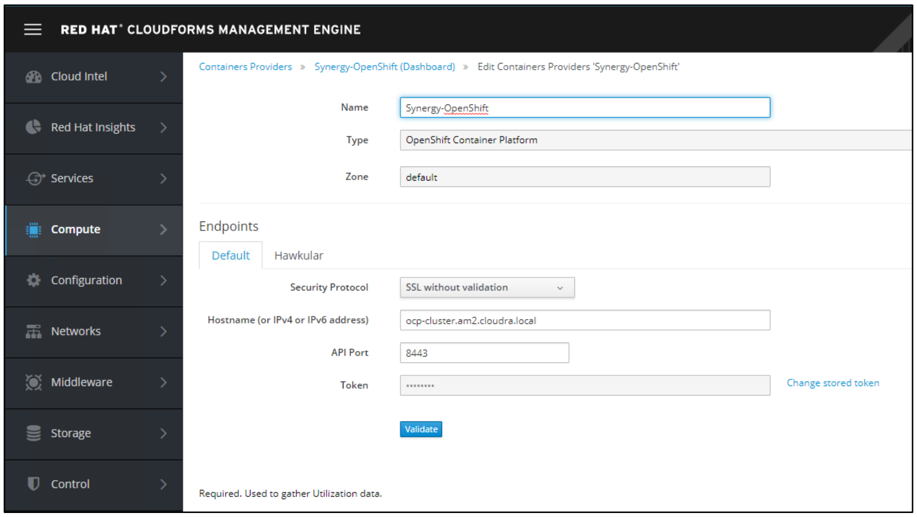


Figure 29: Configure OpenShift provider in CloudForms

Select the Hawkular tab and specify the hostname and port as shown in Figure 30 and click Validate and then Save.

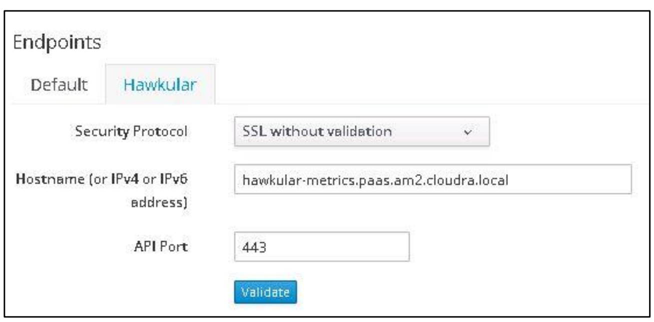


Figure 30: Hawkular metrics collection

An overview of the OpenShift Container Platform will be displayed after the Container Provider makes a successful connection. As CloudForms collects the metrics from the OpenShift provider, updated statistics are provided as seen in Figure 31.

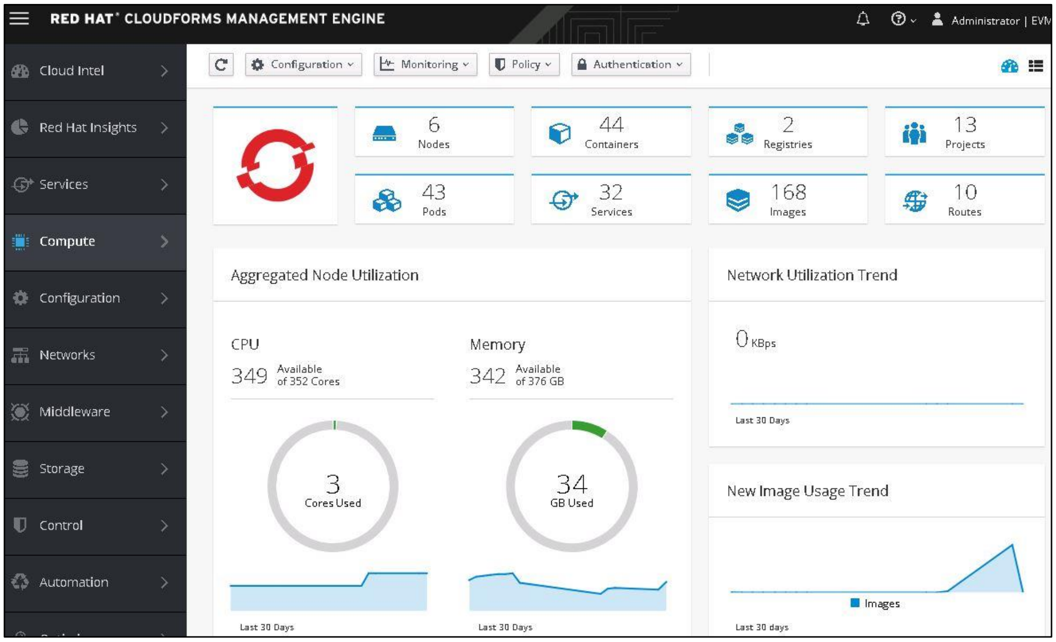


Figure 31: OpenShift provider dashboard

CloudForms also provides chargeback and billing reports for the Red Hat OpenShift Container Platform based services. Enabling billing and chargeback for the OpenShift cluster requires creating a new charge rate that uses the OpenShift Container Platform Provider created previously. To create a new chargeback rate select Cloud Intel > Chargeback > Rates > Compute > Configuration > Add a new Chargeback Rate

Provide a description of the new chargeback rate, select the currency and configure the chargeback rate details, then select Add to save the new chargeback rate.

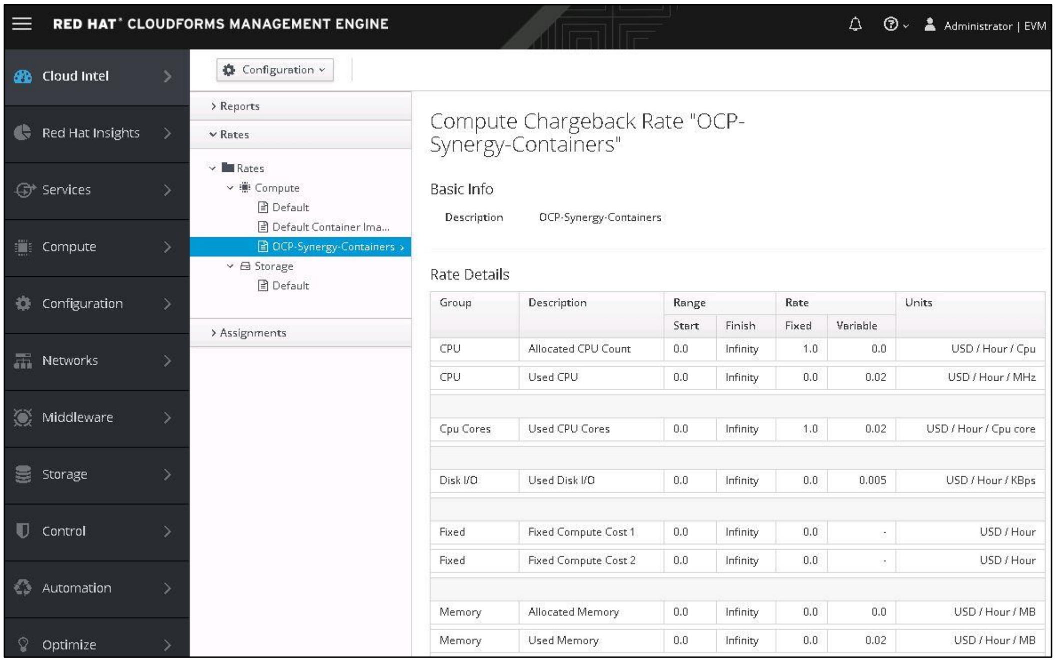


Figure 32: Creating Chargeback reports

A chargeback report can now be created. Select Cloud Intel > Reports > Configuration > Add a new Report. Provide a Menu Name and Title. In the Configure Report Columns section, select Chargeback Container Images from the drop down menu. Select the fields to be included in the report. In this example, the following fields were selected:

- CPU Cores Used Cost
- Memory Used Cost
- Network I/O Used Cost
- Total Cost

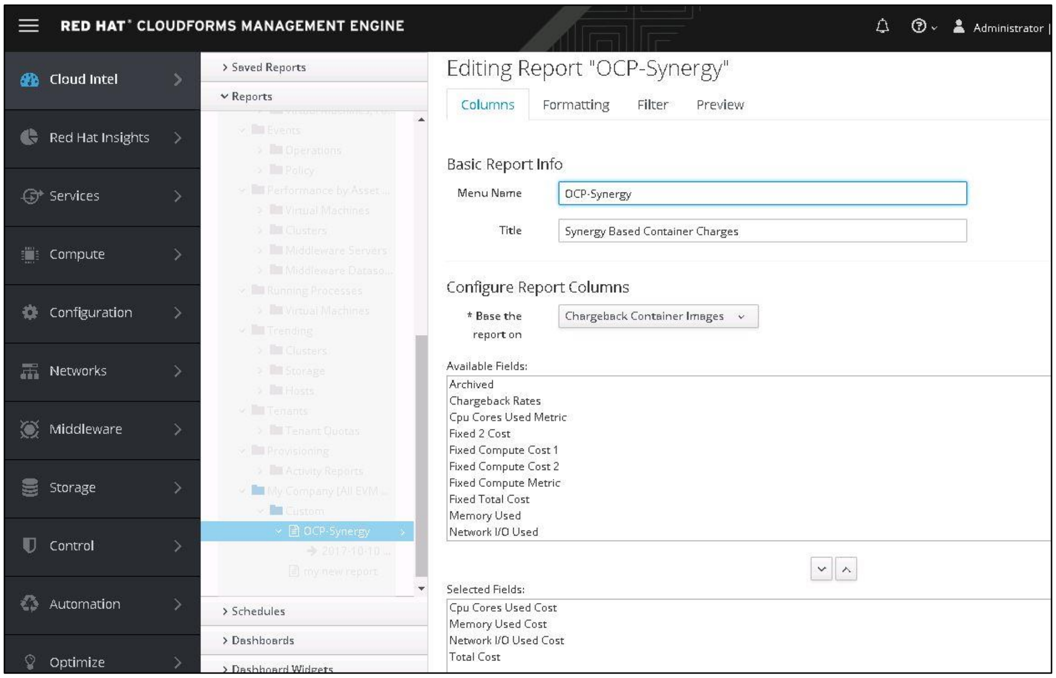
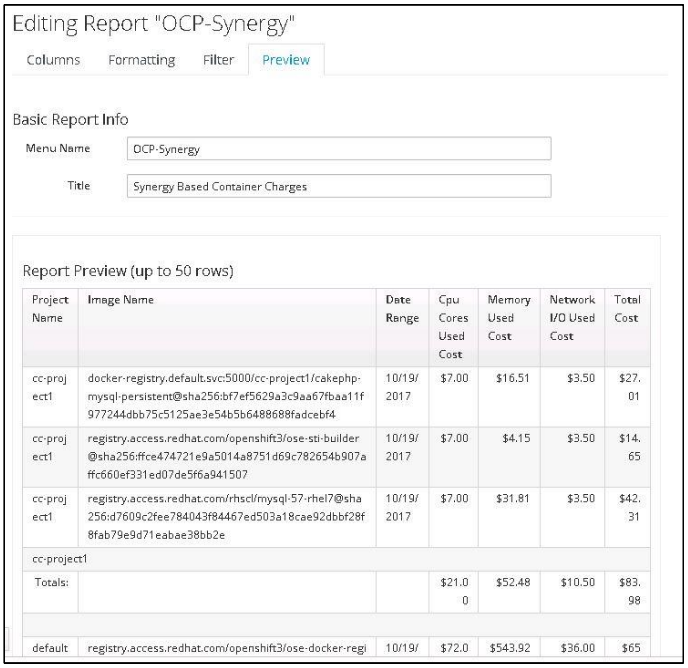


Figure 33: Creating custom reports

From the Filter tab, select Show Costs by Container Image and specify your provider and select the container images you want in your report. Select the Preview tab to generate a preview of the report as shown in Figure 34.



Editing Report "OCP-Synergy"

Columns Formatting Filter Preview

Basic Report Info

Menu Name: OCP-Synergy

Title: Synergy Based Container Charges

Report Preview (up to 50 rows)

Project Name	Image Name	Date Range	Cpu Cores Used Cost	Memory Used Cost	Network I/O Used Cost	Total Cost
cc-project1	docker-registry.default.svc:5000/cc-project1/cakephp-mysql-persistent@sha256:bf7ef5629a3c9aa67fbaa11f977244dbb75c5125ae3e54b5b6488688fadcebf4	10/19/2017	$7.00	$16.51	$3.50	$27.01
cc-project1	registry.access.redhat.com/openshift3/ose-sti-builder@sha256:ffce474721e9a5014a8751d69c782654b907affc660ef331ed07de5f6a941507	10/19/2017	$7.00	$4.15	$3.50	$14.65
cc-project1	registry.access.redhat.com/rhscl/mysql-57-rhel7@sha256:d7609c2fee784043f84467ed503a18cae92dbbf28f8fab79e9d71eabae38bb2e	10/19/2017	$7.00	$31.81	$3.50	$42.31
cc-project1						
Totals:			$21.00	$52.48	$10.50	$83.98
default	registry.access.redhat.com/openshift3/ose-docker-regi	10/19/	$72.0	$543.92	$36.00	$65

Figure 34: Chargeback report preview

Appendix F: Deploying a sample application

This section will demonstrate how to create a project using an existing Red Hat Reference Configuration JBoss EAP 7 microservices on OpenShift Container Platform. The sample application is deployed from one of the Red Hat OpenShift Container Platform Master nodes.

1. Log in to the first Master node as root user.
2. Log in to the Red Hat OpenShift Container Platform as an administrator

   ```
   oc login -u system:admin
   ```

3. Create a new project and set the admin to an existing OpenShift user account. In this example we used an account named "jdoe"

   ```
   oadm new-project msa --display-name="OpenShift 3 MSA on EAP 7" --description="This is a sample
   microservices architecture built on JBoss EAP 7" --admin=jdoe
   ```

4. Log in as the user.

   ```
   oc login -u jdoe
   ```

5. Create the database pods.

   ```
   oc new-app -e MYSQL_USER=product -e MYSQL_PASSWORD=password -e MYSQL_DATABASE=product -e
   MYSQL_ROOT_PASSWORD=passwd mysql --name=product-db
   ```

   ```
   oc new-app -e MYSQL_USER=sales -e MYSQL_PASSWORD=password -e MYSQL_DATABASE=sales -e
   MYSQL_ROOT_PASSWORD=passwd mysql --name=sales-db
   ```

oc status

6. Verify the JBoss EAP image streams

```
oc login -u system:admin
oc project openshift
oc get imagestreams
oc login -u jdoe
```

7. Create the billing service

```
oc new-app jboss-eap70-openshift~https://github.com/RHsyseng/MSA-EAP7-OSE.git --context-dir=Billing
--name=billing-service
oc get all
oc get builds
```

8. Create the JBoss EAP product-service and sales-service applications

```
oc new-app -e MYSQL_USER=product -e MYSQL_PASSWORD=password jboss-eap70-
openshift~https://github.com/RHsyseng/MSA-EAP7-OSE.git --context-dir=Product --name=product-service
oc new-app -e MYSQL_USER=sales -e MYSQL_PASSWORD=password jboss-eap70-
openshift~https://github.com/RHsyseng/MSA-EAP7-OSE.git --context-dir=Product --name=sales-service
```

9. Create the presentation application

```
oc new-app jboss-eap70-openshift~https://github.com/RHsyseng/MSA-EAP7-OSE.git --context-
dir=Presentation --name=presentation
oc status
```

10. Expose the presentation service

```
oc expose service presentation --hostname=msa.paas.cloudra.local
```

11. Access the application from a browser

```
http://msa.paas.cloudra.local/demo.jsp
```

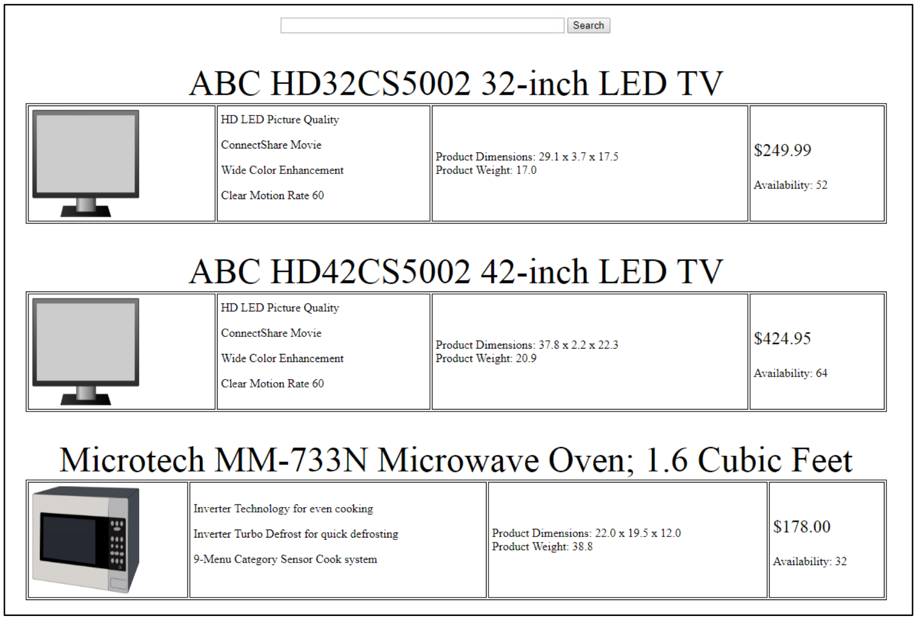


Figure 35: Sample application deployed on OpenShift Container Platform

Once this service is deployed, it is automatically monitored by Red Hat CloudForms. The JBoss EAP 7 application can be viewed under Compute > Containers > Projects > msa. The Summary can be seen in Figure 36.

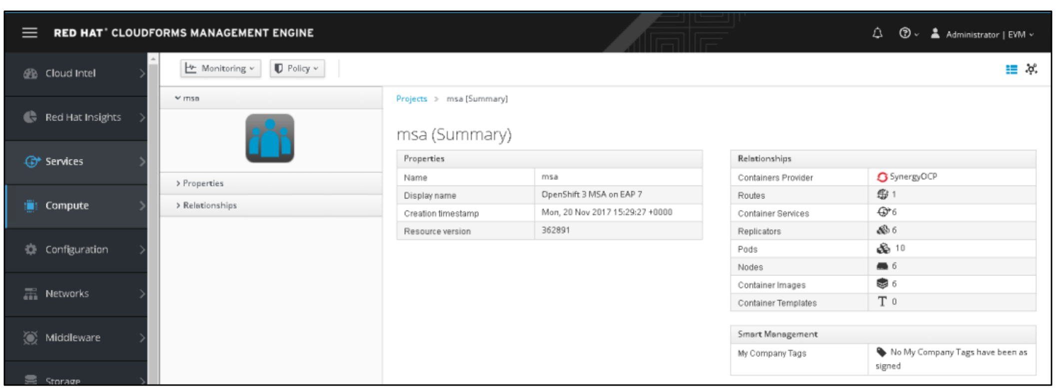


Figure 36: JBoss EAP 7 msa Project Summary

Additional performance metrics about the application can be view by selecting Capacity and Utilization, here we can see statistics on CPU, Memory, and Network utilization.

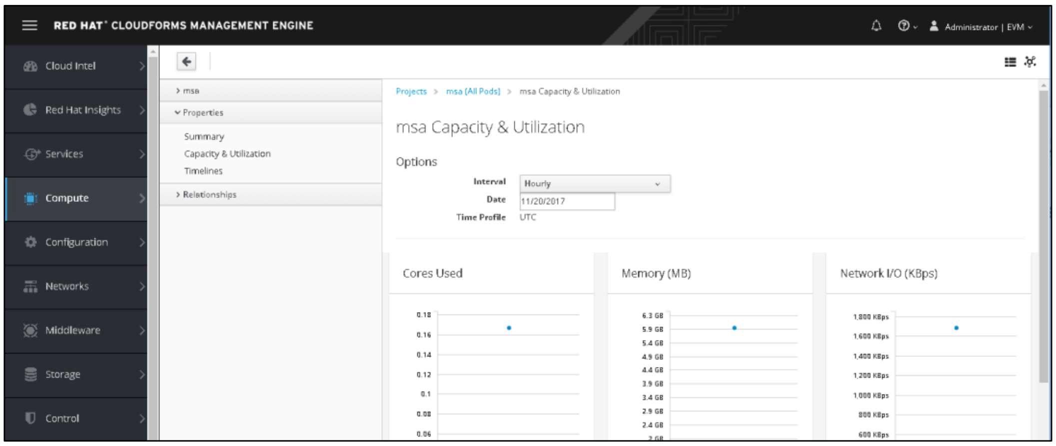


Figure 37: JBoss EAP 7 Capacity and Utilization information

Details about the JBoss EAP 7 container deployment can be view by selecting Relationships. In **Figure 38** the Pods associated with the JBoss EAP 7 application deployment are displayed.

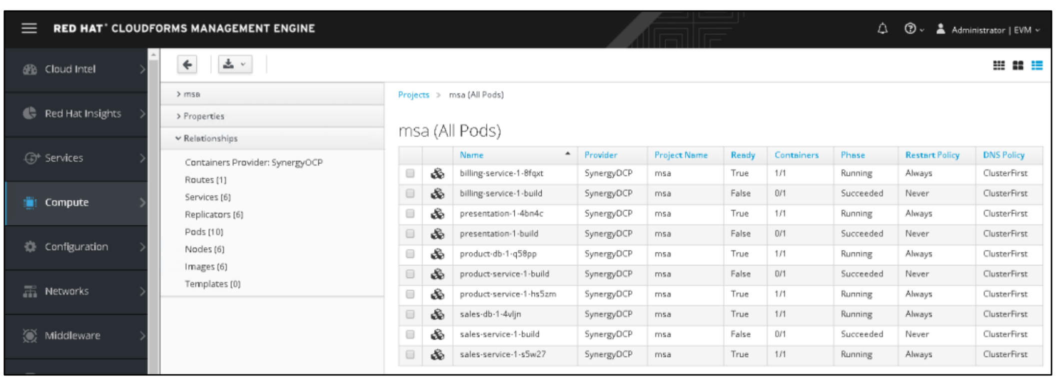


Figure 38: JBoss EAP 7 Pods

Appendix G: Ansible playbooks and customization

Ansible Playbooks

- cleanup

  The cleanup playbook removes the servers from Satellite and HPE OneView. Although not necessary on an initial run, having such a playbook in a workflow is handy to facilitate a CI/CD (continuous integration/delivery) environment. When code is pushed to the git repo, or newer versions of the RPMs are available for testing, the entire workflow can be executed again, with the first step ensuring a clean build environment.

- provisioning

  The provisioning playbook uses the Ansible Modules for OneView to deploy all the servers needed for the solution. This playbook is responsible for creating the server profiles, assigning the profile to a server and powering on the server.

- predeploy

  When deploying a complex infrastructure such as Red Hat OpenShift Container Platform, there are usually some prerequisite steps to prepare the environment, before installing the software itself. The prerequisite tasks for this OpenShift Reference Configuration are captured in the hp_predeploy role.

- openshift

  The official Red Hat OpenShift Container Platform ansible playbooks were used for this Reference Configuration. Detailing these playbooks is out of scope for this document. The playbooks can be found on github and instructions on how to use them can be found here. The playbooks require no modification to work with Ansible Tower. As per the installation guide, a host file is created and a number of variables are defined.

- postdeploy

  This playbook has tasks that create the CloudForms service account (cfadmin) and assigns the OpenShift cluster-reader access role to the cfadmin service account. This playbook also configures glusterfs as the default storage class for the OpenShift cluster.

- add_to_oneview

  Adds a new node to OneView and Ansible Inventory

- fix_inventory

  Removes node from new_nodes group

- scaledown

  Removes a node from HPE OneView and Red Hat OpenShift Container Platform cluster.

- scaleup

  Adds an additional HPE Synergy compute module to the Red Hat OpenShift Container Platform cluster

Customizing the inventory and variables

Customize the inventory file on the ansible host. The node names, IP addresses, and installation location (enclosure/bay) are set in the `/etc/ansible/ocp-on-synergy/hosts` file as shown below. Specifying the exact location for deployment of each node during initial installation is needed to insure that one member of each component is deployed in each of the 3 frames for high availability. Additional nodes can be deployed on any available server.

The container-native storage nodes have an additional configuration in the gluster_fs section. The gluster_fsdevices specification indicates the device as seen by the RHEL 7.4 OS to use for gluster storage. Note that this location (/dev/sdb) can vary depending on whether or not local storage is present and being used for container deployments.

```
 [tower]
ansible.example.com

[OSEv3:children]
nodes
masters
etcd
lb
glusterfs
nfs
new_nodes

[nodes]
```

```
ocp-master1.example.com  openshift_schedulable=False ansible_host=192.168.1.112 enclosure_name='FRM1-TOP: CN1234567A' enclosure_bay=1
ocp-master2.example.com  openshift_schedulable=False ansible_host=192.168.1.113 enclosure_name='FRM2-MID: CN1234567B' enclosure_bay=1
ocp-master3.example.com  openshift_schedulable=False ansible_host=192.168.1.114 enclosure_name='FRM3-BTM: CN1234567C' enclosure_bay=1
ocp-infra1.example.com  openshift_node_labels="{'region': 'infra'}" ansible_host=192.168.1.115 enclosure_name='FRM1-TOP: CN1234567A' enclosure_bay=2
ocp-infra2.example.com  openshift_node_labels="{'region': 'infra'}" ansible_host=192.168.1.116 enclosure_name='FRM2-MID: CN1234567B' enclosure_bay=2
ocp-infra3.example.com  openshift_node_labels="{'region': 'infra'}" ansible_host=192.168.1.117 enclosure_name='FRM3-BTM: CN1234567C' enclosure_bay=2
ocp-cns1.example.com  ansible_host=192.168.1.118 enclosure_name='FRM1-TOP: CN1234567A' enclosure_bay=7
ocp-cns2.example.com  ansible_host=192.168.1.119 enclosure_name='FRM2-MID: CN1234567B' enclosure_bay=7
ocp-cns3.example.com  ansible_host=192.168.1.120 enclosure_name='FRM3-BTM: CN1234567C' enclosure_bay=7
ocp-node1.example.com  ansible_host=192.168.1.121 enclosure_name='FRM1-TOP: CN1234567A' enclosure_bay=8
ocp-node2.example.com  ansible_host=192.168.1.122 enclosure_name='FRM2-MID: CN1234567B' enclosure_bay=8
ocp-node3.example.com  ansible_host=192.168.1.123 enclosure_name='FRM3-BTM: CN1234567C' enclosure_bay=8

[masters]
ocp-master1.example.com
ocp-master2.example.com
ocp-master3.example.com

[infra]
ocp-infra1.example.com
ocp-infra2.example.com
ocp-infra3.example.com

[etcd]
ocp-master1.example.com
ocp-master2.example.com
ocp-master3.example.com

[lb]
ocp-lbnfs.example.com ansible_host=192.168.1.111 enclosure_name='FRM3-BTM: CN1234567C' enclosure_bay=11
[nfs]
ocp-lbnfs.example.com

[glusterfs]
ocp-cns1.example.com glusterfs_ip=192.168.1.118  glusterfs_devices='["/dev/sdb"]'
ocp-cns2.example.com glusterfs_ip=192.168.1.119  glusterfs_devices='["/dev/sdb"]'
ocp-cns3.example.com glusterfs_ip=192.168.1.120  glusterfs_devices='["/dev/sdb"]'

[new_nodes]
```

Additional variables used by ansible are configured in the group_vars/all file found in /etc/sysconfig/ansible/ocp-on-synergy. This file contains the information the ansible playbooks use to deploy the server profiles on the HPE Synergy compute modules. The contents of the file as it was used for testing is shown below.

```
---
master_deployment_plan: "HPE-RHEL7.4-OpenShiftMaster"
worker_deployment_plan: "HPE-RHEL7.4-OpenShiftWorker"
cns_deployment_plan: "HPE-RHEL7.4-OpenShiftCNS"
infra_deployment_plan: "HPE-RHEL7.4-OCP-Infra"
lbnfs_deployment_plan: "HPE-RHEL7.4-OpenShiftLBNFS"
master_server_profile_template: "OCP-Master"
worker_server_profile_template: "OCP-Worker"
cns_server_profile_template: "OCP-CNS-Node"
```

```
lbnfs_server_profile_template: "OCP-LB-NFS"
infra_server_profile_template: "OCP-Infra"
deployment_plan: "{{ worker_deployment_plan }}"
server_profile_template: "{{ worker_server_profile_template }}"
## ocp
openshift_master_default_subdomain: paas.example.com
openshift_master_cluster_hostname: ocp-cluster.example.com
openshift_master_cluster_public_hostname: ocp-cluster.example.com
## oneview vars
oneview_auth:
  ip: "192.168.1.10"
  username: "Administrator"
  api_version: 300
## network settings
network:
  name: "MGMT1"
  gateway: "192.168.1.1"
  netmask: "255.255.255.0"
  domain: "example.com"
  dns1: "192.168.1.106"
  dns2: "192.168.1.104"
dns_server: "{{ network.dns1 }}"
## satellite
satellite_user: "admin"
satellite_url: "https://192.168.1.106"
inventory_path: "/etc/ansible/ocp-on-synergy"
```

Ansible vault

These playbooks use ansible-vault to encrypt sensitive login credentials. Modify passwords.yaml to reflect the user names and passwords used in your environment and vault it using the commands shown below:

```
$ ansible-vault encrypt passwords.yml --output=roles/passwords/vars/passwords.yml
$ chmod 644 roles/passwords/vars/passwords.yml
```

Appendix H: Red Hat OpenShift Container Platform deployment validation and troubleshooting

Post Deployment Verification

This section provides post deployment and validation information.

Verify Red Hat OpenShift Container Platform ETCD cluster is running

To verify the Red Hat OpenShift Container Platform deployment was successful check the ectd cluster health by logging into the first master node and running the following command:

```
etcdctl -C https://ocp-master1.cloudra.local:2379,https://ocp-master2.cloudra.local:2379,https://ocp-master3.cloudra.local:2379\
--ca-file=/etc/origin/master/master.etcd-ca.crt \
--cert-file=/etc/origin/master/master.etcd-client.crt \
--key-file=/etc/origin/master/master.etcd-client.key \
cluster-health
```

The result of the `etcdctl cluster-health` command is shown below, indicating the cluster is healthy.

```
[root@ocp-master1 ~]# etcdctl -C https://ocp-master1.cloudra.local:2379,https://ocp-master2.cloudra.local:2379,https://ocp-master3.cloudra.local:2379 --ca-file=/etc/origin/master/master.etcd-ca.crt --cert-file=/etc/origin/master/master.etcd-client.crt --key-file=/etc/origin/master/master.etcd-client.key cluster-health
```

```
member 3680e0da3823679d is healthy: got healthy result from https://10.60.50.114:2379
member 79cffbdec6e32038 is healthy: got healthy result from https://10.60.50.113:2379
member e8db49bf13a3e053 is healthy: got healthy result from https://10.60.50.112:2379
cluster is healthy
```

List nodes

Executing the `oc get nodes` command will display the OpenShift Container Platform nodes and their respective status. Notice that scheduling is disabled on the ocp master nodes.

```
[root@ocp-master1 ~]# oc get nodes
NAME                          STATUS                     AGE       VERSION
ocp-cns1.cloudra.local      Ready                      2d        v1.6.1+5115d708d7
ocp-cns2.cloudra.local      Ready                      2d        v1.6.1+5115d708d7
ocp-cns3.cloudra.local      Ready                      2d        v1.6.1+5115d708d7
ocp-infra1.cloudra.local    Ready                      2d        v1.6.1+5115d708d7
ocp-infra2.cloudra.local    Ready                      2d        v1.6.1+5115d708d7
ocp-infra3.cloudra.local    Ready                      2d        v1.6.1+5115d708d7
ocp-master1.cloudra.local   Ready,SchedulingDisabled   2d        v1.6.1+5115d708d7
ocp-master2.cloudra.local   Ready,SchedulingDisabled   2d        v1.6.1+5115d708d7
ocp-master3.cloudra.local   Ready,SchedulingDisabled   2d        v1.6.1+5115d708d7
ocp-node1.cloudra.local     Ready                      2d        v1.6.1+5115d708d7
ocp-node2.cloudra.local     Ready                      2d        v1.6.1+5115d708d7
ocp-node3.cloudra.local     Ready                      2d        v1.6.1+5115d708d7
```

List projects

Using the `oc get projects` command will display all projects that the user has permission to access.

```
[root@ocp-master1 ~]# oc get projects
NAME                     DISPLAY NAME              STATUS
default                                                          Active
glusterfs                                                        Active
kube-public                                              Active
kube-system                                              Active
logging                                                          Active
management-infra                                         Active
msa                OpenShift 3 MSA on EAP 7       Active
openshift                                        Active
openshift-infra                                          Active
```

List pods

The `oc get pods` command will list the running pods in the current project. The command below shows the pods for the default project.

```
[root@ocp-master1 ~]# oc get pods
NAME                             READY     STATUS    RESTARTS   AGE
docker-registry-1-9h77j    1/1       Running            0          2d
docker-registry-1-b4fd9    1/1       Running            0          2d
docker-registry-1-jt7zs      1/1       Running            0          2d
registry-console-1-rgb40   1/1       Running           0          2d
router-1-bkztq                   1/1       Running           0          2d
router-1-wlj69                   1/1       Running           0          2d
router-1-zmp9k                 1/1       Running           0          2d
```

Verify the Red Hat OpenShift Container Platform User Interface

Log into the Red Hat OpenShift Container Platform user interface using the public url defined in the openshift_master_cluster_public_hostname variable as shown below.

`openshift_master_cluster_public_hostname: ocp-cluster.cloudra.local`

The login URL in this example is https://ocp-cluster.cloudra.local:8443/console/.

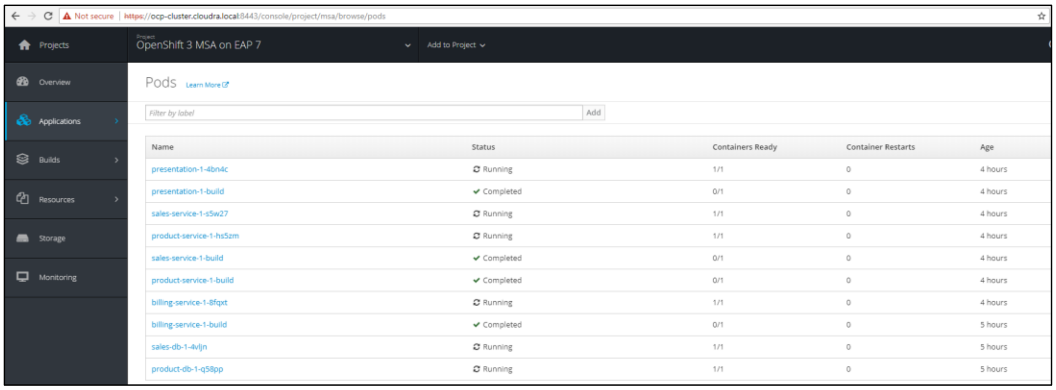


Figure 39: Red Hat OpenShift Container Platform portal

Resources and additional links

HPE Reference Architectures, hpe.com/info/ra

HPE GitHub Reference Architectures, https://github.com/HewlettPackard/image-streamer-reference-architectures

HPE Servers, hpe.com/servers

HPE Storage, hpe.com/storage

HPE Networking, hpe.com/networking

HPE Synergy, hpe.com/synergy

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

Red Hat Ansible Tower, http://docs.ansible.com/ansible-tower/

Red Hat Gluster Storage, https://access.redhat.com/documentation/en/red-hat-gluster-storage/

Red Hat OpenShift Container Platform, https://access.redhat.com/documentation/en/openshift-container-platform/

Red Hat Satellite, https://access.redhat.com/documentation/en/red-hat-satellite/

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

Sign up for updates

4AA4-xxxxENW, Month 20XX, Rev. #