Hewlett Packard
Enterprise

HPE Reference Configuration for Kubernetes, Portworx PX-Enterprise on HPE Synergy

Reference Architecture

Contents

Executive summary

As organizations move to adopt containers and microservices, users are faced with what types of workloads to containerize and what types of containers to deploy in production. With the right infrastructure, stateful containers improve business agility and IT operational experience in the same manner as stateless container workloads. For enterprises, the benefits include enabling developer agility, improving resource usage, and more efficient scaling within IT managed environments.

This Reference Configuration provides architectural guidance for deploying, scaling, and managing stateful cloud-native applications with Kubernetes as a scale-out and efficient container orchestrator and Portworx PX-Enterprise edition as a cloud-native software-defined storage platform on HPE Synergy composable infrastructure. More specifically, it shows how to:

- Deploy Container-as-a-Service (CaaS) and managed container functionality within IT organizations using Kubernetes as the orchestration engine
- Run stateful workloads using Portworx PX-Enterprise as the cloud-native storage platform
- Automate deployment and scale of an integrated Portworx + Kubernetes platform leveraging HPE Synergy strengths in rapid provisioning using HPE Synergy Image Streamer by orchestrating through Ansible.

Kubernetes provides a state of the art compute platform for containerized applications and enables IT operation teams to bring applications into production quickly and manage applications in production more effectively. Portworx PX-Enterprise is a cloud-native storage product that provides elastic container storage, container data lifecycle management, and complete Kubernetes integration. The combination of Kubernetes and Portworx PX-Enterprise with HPE composable infrastructure allows organizations to effectively run stateful containers in production. The composable nature of HPE Synergy enables IT operators to easily assign and re-assign physical, virtual, or container resources based on workload types and utilization demands. Using the composability and fluid resource pools features from HPE Synergy, organizations can minimize over-provisioning of underlying infrastructure resources and consequently reduce capital expenditure in the process.

Target audience: This document is intended for IT architects, DevOps teams, systems integrators, and partners who are planning to deploy and manage containers on a large scale running on HPE Synergy composable infrastructure with Kubernetes.

Document purpose: The purpose of this document is to describe a best practice scenario for deploying stateful Kubernetes workloads on bare metal HPE Synergy with Portworx as the container data-infrastructure. Cassandra is used as the example of a stateful and a scale-out workload. Readers can use this document to achieve the following goals:

- Gain insight into the value proposition for running Kubernetes container workloads in bare metal environments leveraging the strengths of the HPE Synergy composable infrastructure.
- Learn how to rapidly deploy Kubernetes and Portworx PX-Enterprise using HPE Synergy Image Streamer technology.

Introduction

This Reference Configuration describes a Kubernetes-as-container orchestration platform and Portworx as a cloud-native storage platform on HPE composable infrastructure and includes best practices to allow IT teams to bring up new services within minutes using infrastructure automation and deployment tools such as Ansible.

The HPE Synergy platform is designed to co-host traditional and cloud-native applications with the implementation of composable infrastructure. Composable infrastructure combines the use of fluid resource pools, made up of compute, storage, and fabric with software-defined intelligence. Composable pools of compute, storage, and fabric can be intelligently and automatically combined to support any workload. The resource pools can be flexed to meet the needs of any business application. The use of HPE Synergy Image Streamer in the solution allows for rapid image and application changes to multiple compute nodes in an automated process.

IT organizations are looking to deploy Kubernetes worker nodes on bare metal servers to optimize performance and resource efficiency of servers. By running bare metal servers intelligently provisioned by the HPE Synergy Image Streamer and using containers to isolate applications instead of a hypervisor more applications can be supported per server. This consolidation leads to reduced capex costs by minimizing virtualization and operating system instance licensing costs. Applications built using microservices and requiring high performance data stores are highly suitable to run on bare metal because they are highly distributed, scale horizontally, need quick instantiation, and don't need hypervisor level features. Deploying them on bare metal allows efficient usage of existing capacities on demand, higher server density, and more reactive systems.

While the cost and performance advantages of bare metal worker nodes are quite attractive for running application workloads, the Kubernetes master nodes (see Figure 1) can be deployed as VMs instead of individual physical instances to enable high availability and simplify maintenance. This provides a single, on-premises infrastructure solution for organizations adopting containers for existing applications or new microservices, allowing them to compose and scale applications where some tiers or services can run as VMs while others run as containers.

HPE Synergy provides an ideal platform to allow for rapid deployment of additional Kubernetes / Portworx nodes to meet changing workload demands. The introduction of HPE Synergy Image Streamer provides the capability to automatically deploy new compute modules immediately with true stateless images which integrate server hardware configuration with operating environment images. This allows for seamless expansion and contraction of physical resource utilization providing optimal use of data center infrastructure.

Solution overview

This Reference Configuration provides a solution architected for Kubernetes-as-container orchestration platform with Portworx PX-Enterprise as a cloud-native storage platform for running stateful workloads on HPE composable infrastructure with integration to provide automated Kubernetes cluster setup and container management as shown in Figure 1. This enables IT teams to deploy a scale-out container platform on bare metal using a combination of HPE Synergy composable system, Kubernetes, and Portworx's cloud-native storage platform. The result is a solution that can be rapidly deployed in 30 minutes or less, is highly container and storage elastic, and can scale to many nodes with bare metal performance.

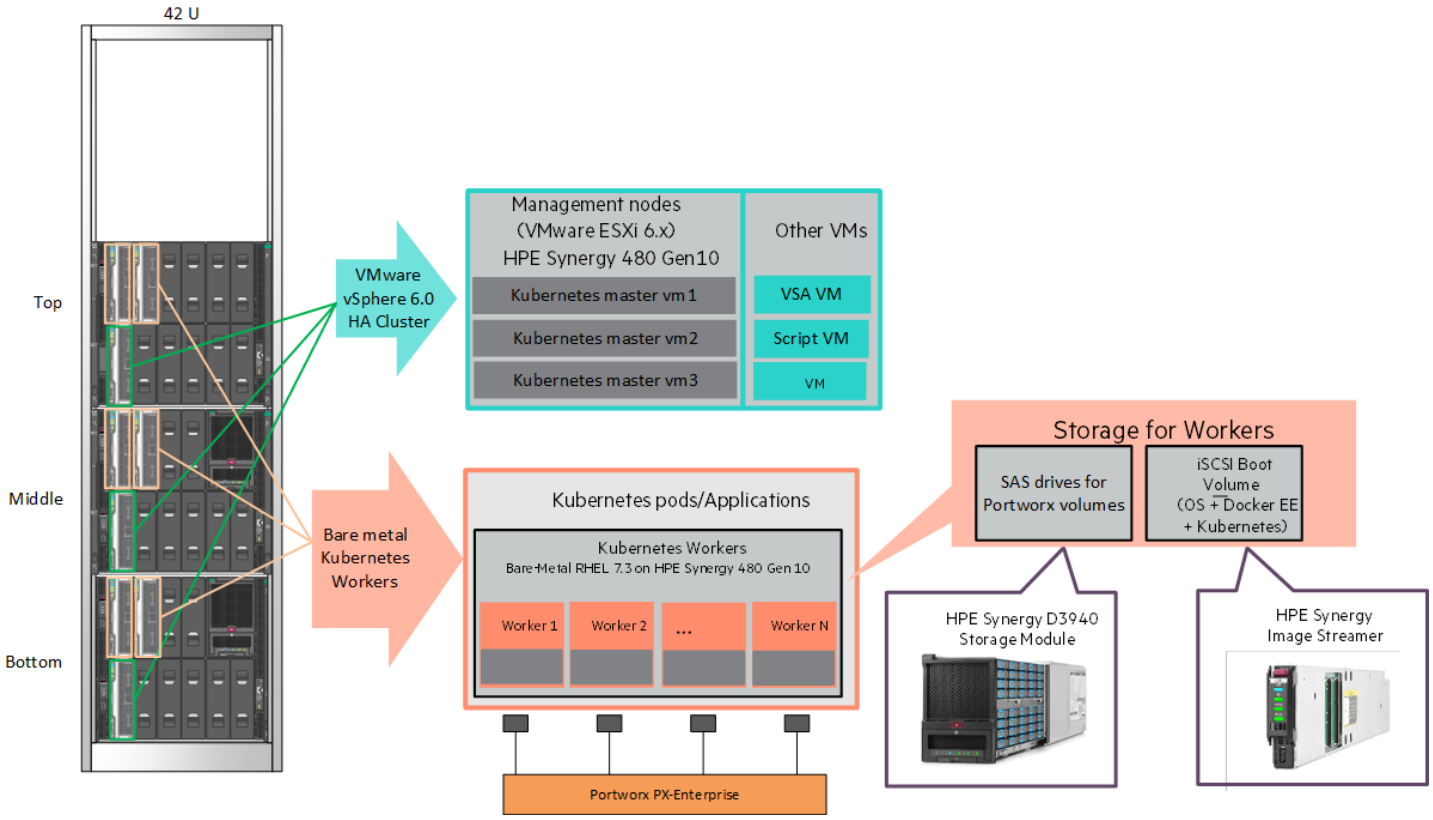

Figure 1. Kubernetes with Portworx PX-Enterprise on HPE Synergy

HPE composable infrastructure provides a scalable platform for stateful container workloads. This solution uses Portworx PX-Enterprise with Kubernetes. The Kubernetes worker nodes and Portworx PX-Enterprise run bare metal on the HPE Synergy 480 Gen10 Compute Modules. Kubernetes creates a compute cluster for containers over the HPE Synergy 480 Gen10 Compute Modules. Portworx PX-Enterprise turns each Kubernetes node into a storage capable node providing high availability at the container data level.

In Figure 1, all three HPE Synergy 12000 Frames are connected by a high-speed backplane link (40 Gbps). Within two of the Synergy 12000 Frames, an HPE Synergy D3940 Storage Module provides SAS storage to the HPE Synergy 480 Gen10 Compute Modules. Portworx PX-Enterprise fingerprints each storage drive for I/O capability and then aggregates them into container-level storage pools for use by Kubernetes. These storage pools are then managed by PX-Enterprise. The top frame does not include an HPE Synergy D3940 Storage Module. The PX-Enterprise presents storage to the compute modules in the top frame using storage pools from the D3940 modules in the middle and bottom frames.

The compute modules outlined in green in Figure 1 are VMware® ESXi nodes. These ESXi nodes use their local disks to form a shared datastore for the ESXi cluster. The datastore is managed by HPE StoreVirtual VSA software. Optionally, the configuration could utilize an external storage array, such as an HPE 3PAR array, to provide storage for the ESXi shared datastores.

Application containers will consume storage using Portworx PX-Enterprise. As shown in this Reference Configuration, PX-Enterprise will allocate storage from the HPE Synergy D3940 Storage Module using a SAS connection. For servers without a D3940 Storage Module available locally in the Synergy 12000 Frame, those servers can use Portworx to remotely access storage over TCP across the high-speed backplanes. Thereby, applications can have hyperconvergence when performance requires it, such as in database scenarios, and access to remote storage capacity for less latency sensitive cases. The benefit of the network-based storage access model is that more compute cores can access storage simultaneously, as typically required in bulk data processing scenarios, making it an extreme case of scaling storage and compute independently.

The flexible storage configurations will be centrally visible and managed by an infrastructure administrator. As applications are launched by Kubernetes, the administrator will have global visibility of where applications are consuming persistent volumes. As an example, the Portworx Lighthouse management console, shown in Figure 2. Portworx Lighthouse management console, can display Cassandra volumes along with other workloads in that cluster.

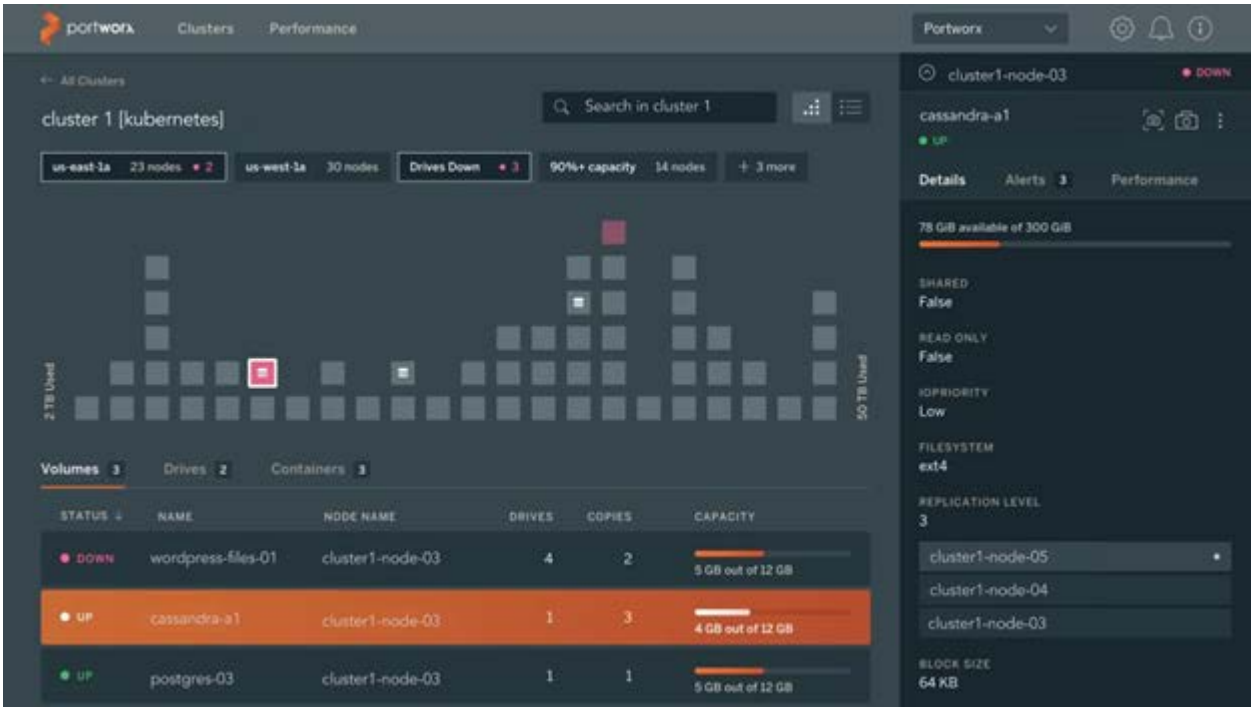

Figure 2. Single cluster view on Portworx Lighthouse management console

The infrastructure administrator also has the ability to view the total amount of storage across a set of clusters from the Portworx Lighthouse management console. This means that performance and configuration management are possible at scale. See Figure 3.

Figure 3. Multi-cluster view from Portworx Lighthouse management console

Once PX-Enterprise is configured, each compute node in the cluster will be capable of running stateful container workloads. Kubernetes dynamically creates pods (encapsulating one or more containers) and persistent volumes. The dynamic creation of persistent volumes is through the PX-Enterprise driver[1] for Kubernetes. Within the Kubernetes specification, the IT team can define whether the application requires high availability storage using a Portworx Storage Class[2]. The persistent volumes are then made highly available automatically by PX-Enterprise.

Table 1. Example of Portworx Storage class for Kubernetes

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: px-storageclass
provisioner: kubernetes.io/portworx-volume
parameters:
  repl: "2"
  priority_io: "high"
  snap_interval: "3600"
```

[1] https://github.com/kubernetes/kubernetes/tree/master/examples/volumes/portworx#persistent-volumes
[2] https://docs.portworx.com/scheduler/kubernetes/storage-class.html

The operational experience is then integrated:

- Persistent volumes are dynamically provisioned and container-granular, so that the following can be uniquely specified:
  - Priority: many application containers will run per server. Setting priority allows the Kubernetes spec to prioritize I/O for the more important applications
  - Replication: each volume selects its desired level of protection
  - Snapshots and backups: only that application volume data is captured
- The scheduler is informed of the persistent volume's location, enabling hyperconvergence including failover.
- All node identities are shared by Kubernetes and Portworx, ensuring easy coordination and management.

As end-user load arrives at the containers and servers, additional nodes can be efficiently scaled out using HPE Synergy Image Streamer to rapidly deploy additional servers to the solution. Golden images and automated deployment plans configured and stored in HPE Synergy Image Streamer enable rapid deployment of stateless bare metal workers which immediately join the converged Kubernetes + Portworx cluster as the server is powered on. Additional data services and seamless management make HPE Synergy an optimal platform for deploying Portworx and Kubernetes.

Solution components

The hardware and software components used to build this HPE Reference Configuration for PX-Enterprise with Kubernetes on HPE composable infrastructure are detailed in this section.

Hardware

The following hardware components were utilized in this Reference Configuration.

Table 2. Hardware components

Component	Purpose
HPE Synergy 12000 Frame	Rack enclosure for compute, storage, and network hardware
HPE Synergy Composer	Server-level resource management
HPE Synergy Image Streamer	Software infrastructure deployment
HPE Synergy 480 Gen10 Compute Modules	Hosts for running Kubernetes and Portworx containers
HPE Synergy D3940 Storage Module	Storage for Kubernetes workloads
HPE FlexFabric 5900AF Switch	Top of Rack network connectivity

HPE Synergy

HPE Synergy, the first 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 Composer provides the enterprise-level management to compose and deploy server-level resources to your application needs. This management appliance uses software-defined intelligence with embedded HPE OneView to aggregate compute, storage, and fabric resources in a manner that scales to your servers 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 and rack servers, 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 HPE Synergy Composer.

HPE Synergy Image Streamer

HPE Synergy Image Streamer is a new approach to deploy and update composable infrastructure. This management appliance works with HPE Synergy Composer for fast software-defined control over physical compute modules with operating system, container infrastructure, 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 4 shows an HPE OneView server profile configured to deploy a compute server with Kubernetes worker node. The Server Profile specifies the required networking, storage, and firmware as well as the OS deployment plan details from HPE Synergy Image Streamer. The physical state and deployed software state are maintained separate from the physical compute node. The physical compute node does not need to retain any state.

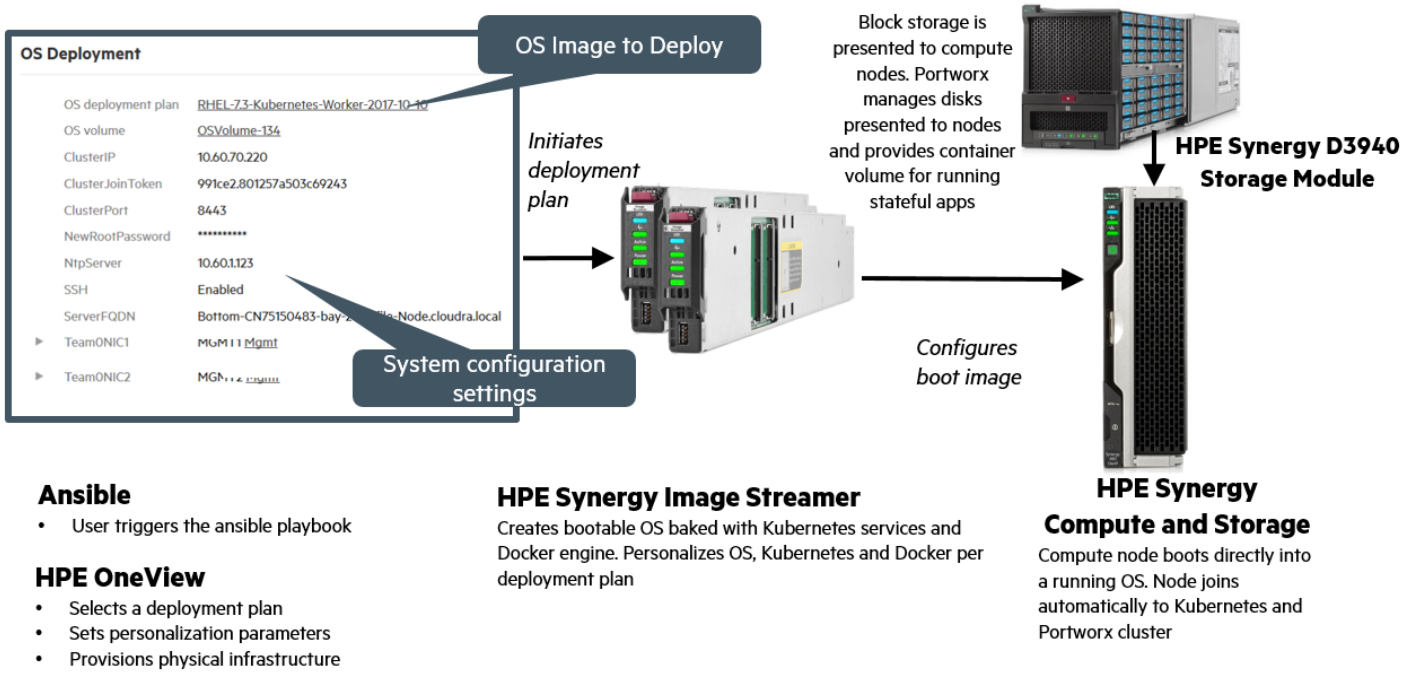

Figure 4. HPE OneView and HPE Synergy Image Streamer

HPE Synergy D3940 Storage Module

The HPE Synergy D3940 Storage Module provides 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. 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 is used for persisting stateful container data. The server profiles for Kubernetes worker nodes include the configuration of local storage managed by PX-Enterprise to create a Portworx volume for containers. Optionally you can consider mounting a Docker or Kubernetes specific volume (*/var/log/docker* or */var/log/kubernetes*) to different storage to minimize the use of limited storage on the HPE Synergy Image Streamer.

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. 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 FlexFabric 5900AF 48XG 4QSFP+ Switch

The HPE FlexFabric 5900 switch series is a family of ultra-low-latency 1/10GbE data center top-of-rack (ToR) switches that is part of the Hewlett Packard Enterprise FlexNetwork architecture HPE FlexFabric solution that is perfect for deployment at the server access layer in large and medium-sized enterprises. These switches can also be used in campus core/distribution layers where higher performance 40GbE connectivity is required with 10GbE links. Virtualized applications and server-to-server traffic require ToR switches that meet the needs for higher-performance server connectivity, convergence, virtual environment support, and low-latency.

Software

The software components used in this Reference Configuration are listed in Table 3 and Table 4.

Table 3. Third-party software

Component	Version
Kubernetes	1.7.4
Portworx PX-Enterprise	1.2.10
Docker EE	17.06

Table 4. HPE management software

Component	Version
HPE Synergy Composer	3.1
HPE Synergy Image Streamer	3.1
HPE Synergy Image Streamer artifacts for Kubernetes and Portworx	https://github.com/HewlettPackard/K8s-Portworx-Synergy/tree/master/imagestreamer

Kubernetes: Production-Grade container orchestration

Kubernetes is a state of the art orchestration system for containers. Born from a decade of experience at Google™ and backed by one of the most vibrant ecosystems, Kubernetes is an effective compute scheduler that enterprises can consider for their cloud-native deployments. Kubernetes provides the necessary tools, platform, and integrations required to run scale-out container workloads. Coupled with Portworx PX-Enterprises, IT teams can run any stateful or stateless container workloads in production.

Kubernetes enables IT teams to:

- Deploy containerized applications with predictability and repeatability
- Scale out and separately update applications quickly
- Control compute resource usage from a central control-plane
- Design operations to be portable across public, private, and hybrid deployments

With HPE Synergy deploying Kubernetes and Portworx, IT can rapidly deploy stateful containers with full-stack hosting capabilities on the HPE platform.

Kubernetes provides the following:

- Centralized container control-plane through the Kubernetes master
- Strong resource isolation for container CPU and memory resources
- Dynamic provisioning of persistent volumes through storage providers such as Portworx
- Desired state managed down to the server node level
- Fine-grained control and visibility through the command-line (kubectl), RESTful API, and Web UI (dashboard)

Figure 5 is an example of the resource visibility and control from the Kubernetes Dashboard. It shows the CPU and memory usage, and status of each Kubernetes worker node. As with the Portworx management console, both GUIs show Kubernetes nodes and pods.

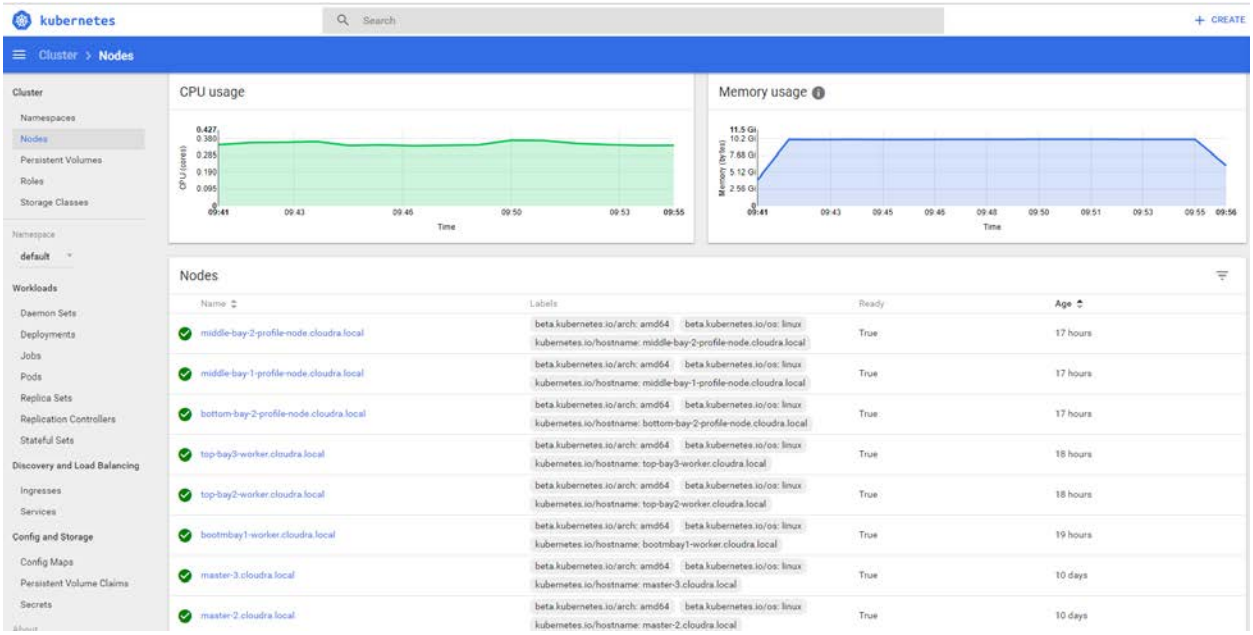

Figure 5. Kubernetes dashboard on HPE Synergy

Also Figure 6 shows Kubernetes pods[3] running on the worker nodes.

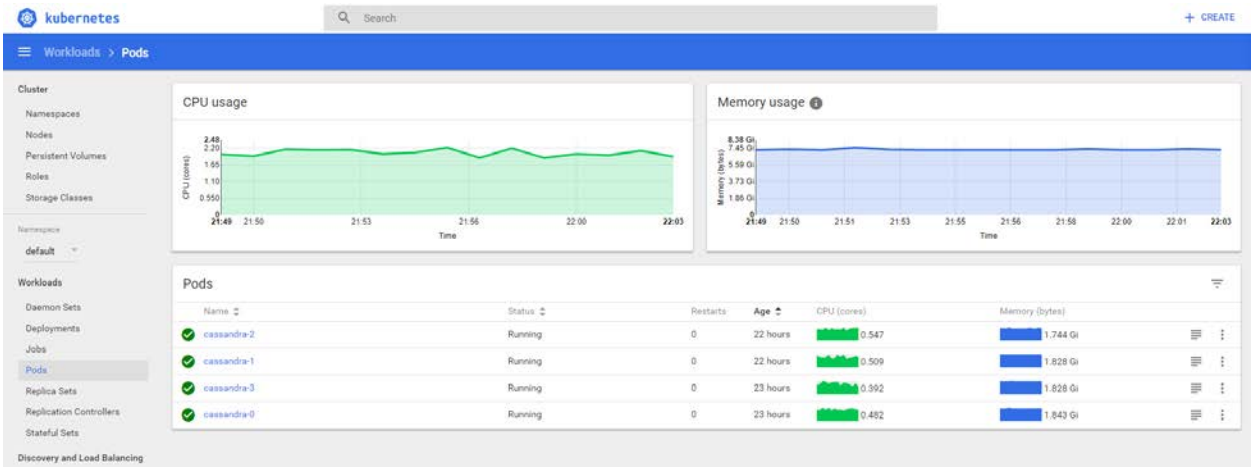

Figure 6. Kubernetes Dashboard

[3] https://kubernetes.io/docs/concepts/workloads/pods/pod/

Portworx PX-Enterprise Metal edition: Cloud-native storage

Portworx PX-Enterprise is a cloud-native storage product that provides the elastic container storage and the container data lifecycle management required for running stateful containers in production. With PX-Enterprise, an IT operator can effectively manage any database or stateful application in any physical or VM infrastructure using any container scheduler. Whether running on-premises or in multiple public clouds, IT teams can effectively adopt stateful microservices in precisely the same manner.

The PX-Enterprise Metal edition is delivered as a container that gets installed on bare metal compute servers. Applications get block storage that is isolated to just that container, expandable, and that can be encrypted. Low latency applications may require that storage be physically local; conversely and to enable burst, compute-heavy applications can run on many servers and remotely access storage over TCP. In this solution, Portworx manages storage as persistent volumes and coordinates with schedulers to enable fast dynamic provisioning, easy high-availability, and full lifecycle management.

Portworx PX-Enterprise Metal features include:

- Elastic container storage
  - Built-in clustering of storage from bare metal servers, VMs, cloud instances, or SANs
  - Block storage that is isolated to a container and managed as persistent volumes
  - Dynamic creation and expansion of those persistent volumes
  - Replication and I/O prioritization at the container-level
  - Parallel file access to a shared volume for NFS-style scenarios
- Container data lifecycle management
  - Snapshots and backups that are automated and integrated with cloud object storage
  - Encryption with Bring Your Own Key (BYOK) for data at rest and in-flight
  - Scanning, alerting, and repair for media errors in physical drives
  - Multi-cluster GUI (Lighthouse management console), CLI, and RESTful APIs
- Schedule-complete integration
  - Deployment and upgrading of Portworx through Kubernetes (as a daemon set)
  - Coordination of application pod scheduling for hyperconvergence, including upon failover
  - Node identities are shared by Kubernetes and Portworx, ensuring easy coordination and management
  - Health checks for Portworx as an infrastructure-level provider

Once running, Portworx can be managed through the command-line, RESTful API, or the GUI called Lighthouse management console. From any of these interfaces, PX-Enterprise enables operators to easily manage persistent volumes and data through tasks like:

- Configuring the BYOK (Bring Your own Key) encryption for container volumes and backups
- Managing and expanding capacity for persistent volumes and/or storage for the cluster
- Setting backup and snapshot configuration for persistent volumes
- Identifying physical drives that require replacement
- Troubleshooting performance of volumes

As an example of managing storage capacity, each server can contribute storage to the cluster. One task that PX-Enterprise makes simple is to see how much storage each server uses. As shown below, Lighthouse includes built-in histograms that arrange servers based on storage and server usage. When a cluster is selected, the servers are shown as boxes and arranged from left to right by storage (Terabytes) used. From this cluster view, an IT operator can further probe and quickly filter to see the persistent volumes based on their I/O priority.

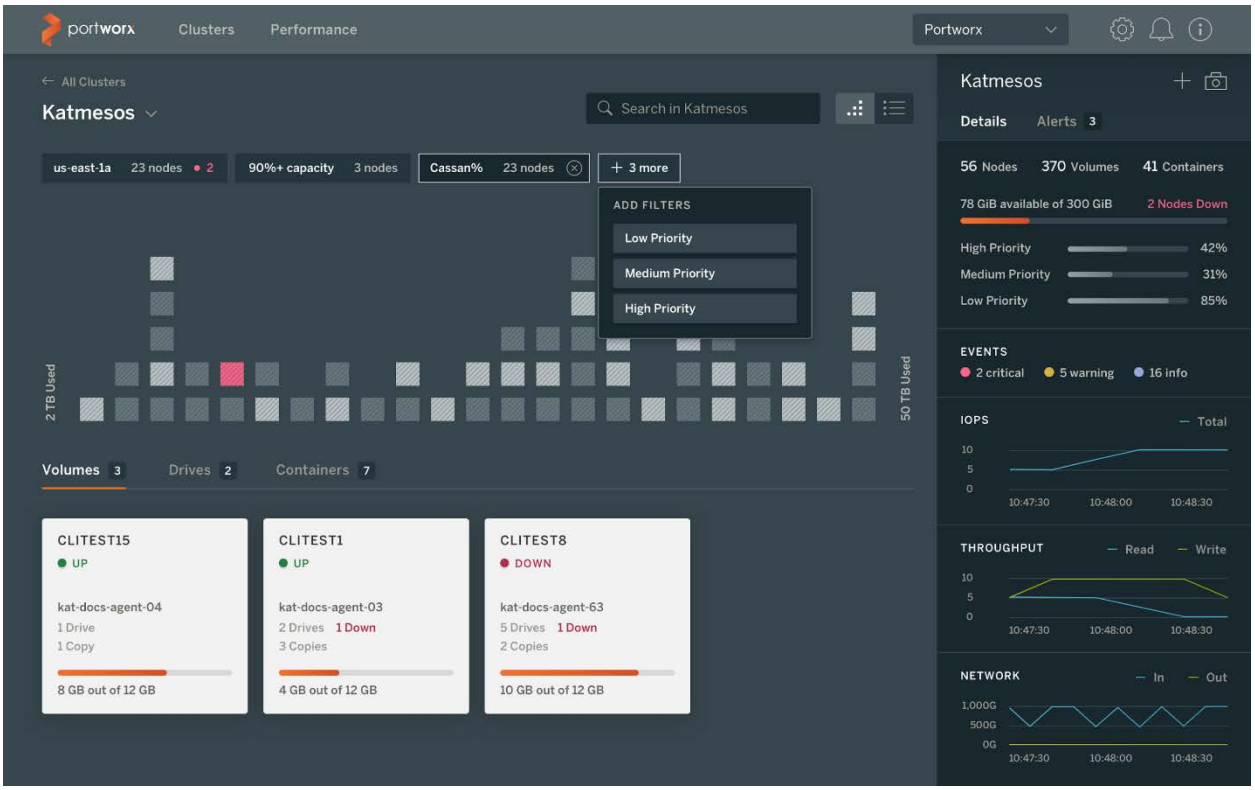

Figure 7. Portworx Lighthouse: inspecting volumes based on filter conditions

Another task can be to find where the persistent volumes are located based on the container image. By selecting the Container view, an IT operator sees the container images running in that cluster. By then clicking on a container image (Cassandra for example), the server nodes with persistent volumes for Cassandra are shown. See Figure 8 where the volume and labels are shown for Cassandra.

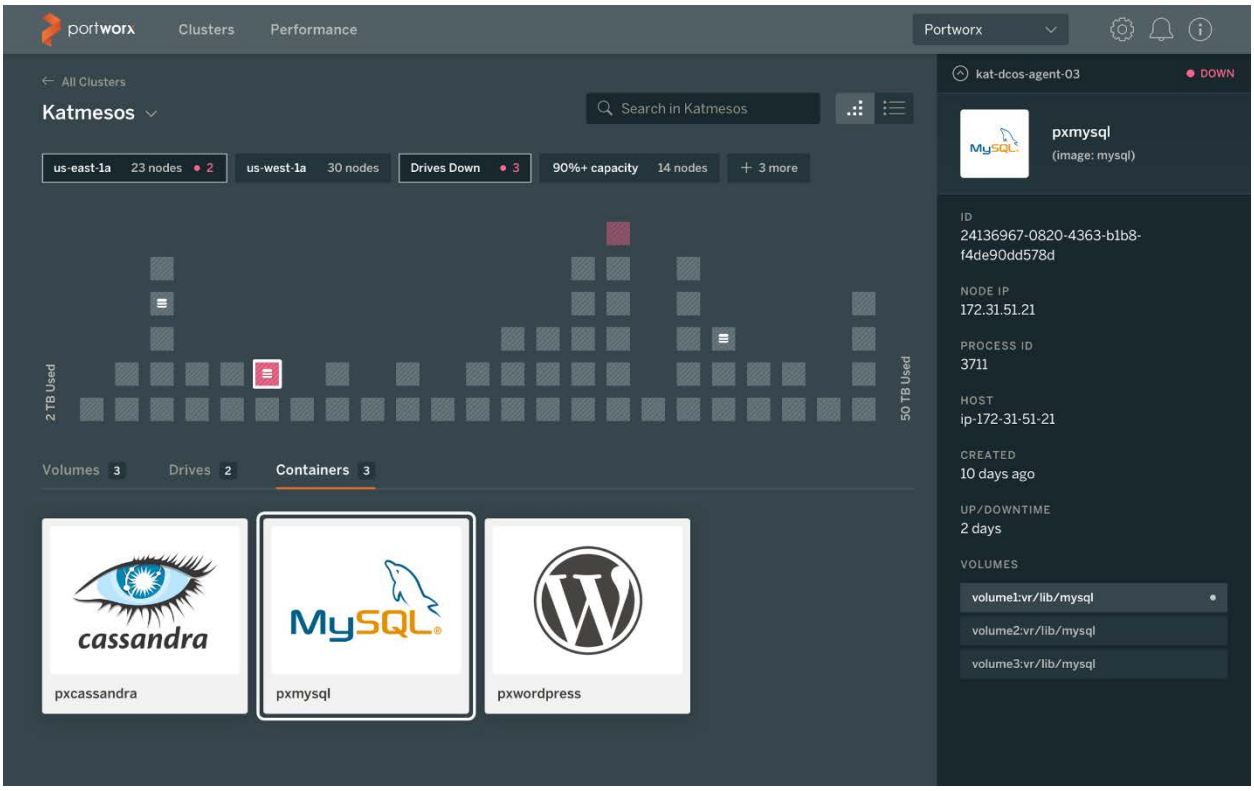

Figure 8. Portworx Lighthouse: identifying volume location based on containers

Portworx PX-Enterprise Metal edition configuration best practices

This section outlines several best practices validated by Hewlett Packard Enterprise as part of this Reference Configuration. HPE recommends following best practices for PX-Enterprise Metal edition deployments, including:

- Configure PX-Enterprise to use separate TCP/IP networks for its data and management traffic. Network separation helps ensure the flow of management traffic during excessive data traffic.
- Verify that the worker nodes with PX-Enterprise have at least 10 Gbps between servers. Having 100 Gbps network connectivity is recommended for more demanding workloads.
- Configure PX-Enterprise CloudSnaps to backup Portworx volumes to an object store. The frequency of backups should be sized to the network connectivity between the cluster and object storage. If unsure, set the backups to an initial frequency of no more than once per day and then increase the frequency.
- Snapshot data volumes prior to any application pod and container upgrades.
- Database and low latency workloads should use local storage (including connecting over SAS to the D3940) when possible.
- Batch workloads can connect over PX-Enterprise's ability to remotely access storage over TCP. Remote access patterns also help add more compute capacity to such workloads.
- As a best practice in production, monitoring should be configured to provide a unified view of metrics for containers' compute and storage. See Appendix D: Monitoring Portworx Volumes.
- The HPE RAID controller can be used to provide hardware assistance. This configuration was tested with RAID-level 5.

Solution setup

In this Reference Configuration, Kubernetes provides the container compute infrastructure, and Portworx PX-Enterprise provides the data infrastructure. HPE Synergy Image Streamer delivers Kubernetes software onto the target servers, along with the operating system. Portworx software is configured and deployed as a Kubernetes daemon set. The Portworx component is not part of the Image Streamer OS deployment plan. Please refer to the Software section for a list of software components used for the Reference Configuration testing. This Reference Configuration concludes with details on how to deploy and run containerized Cassandra as a use case for stateful containers.

This solution is ideal for organizations that develop cloud-native applications for maximizing resource, performance, and cost efficiencies. The worker nodes take advantage of the fluid pool of HPE Synergy bare metal resources. The Kubernetes master nodes are deployed as virtual machines on a VMware ESXi 6.5 set of hosts, as shown in Figure 9. Portworx is deployed as a peer-to-peer system, without a master.

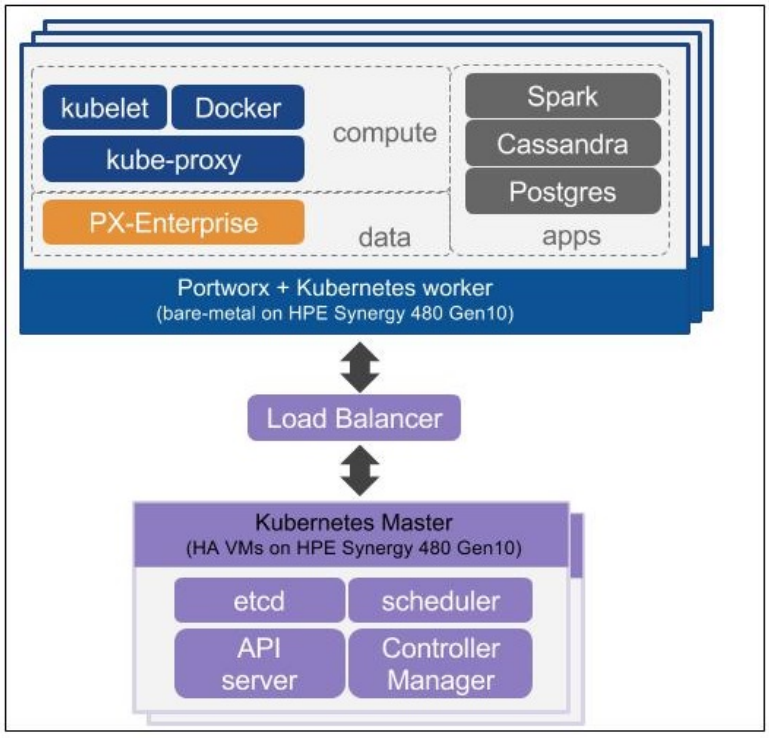

Figure 9. Logical view of master and worker nodes deployed on HPE Synergy

The following is a high-level list of solution setup:

- Setting up the hardware platform (Synergy configuration and automated deployment)
- Setting up the Kubernetes master nodes
- Setting up Kubernetes worker nodes using Image Streamer
- Setting up the Portworx software
- Cassandra NoSQL workload in containers with Kubernetes and Portworx

Each of these is described in the following sections.

Setting up HPE Synergy hardware platform

The three HPE Synergy frames used for this Reference Configuration include HA deployment of HPE Synergy Composer and HPE Synergy Image Streamer. The system layout diagram for this Reference Configuration is shown in Figure 10. In this configuration, two of the Synergy frames have a mix of HPE Synergy 480 Gen10 Compute Modules and Synergy D3940 Storage Module, and one of the frames exclusively hosts the Synergy 480 Gen10 Compute Modules.

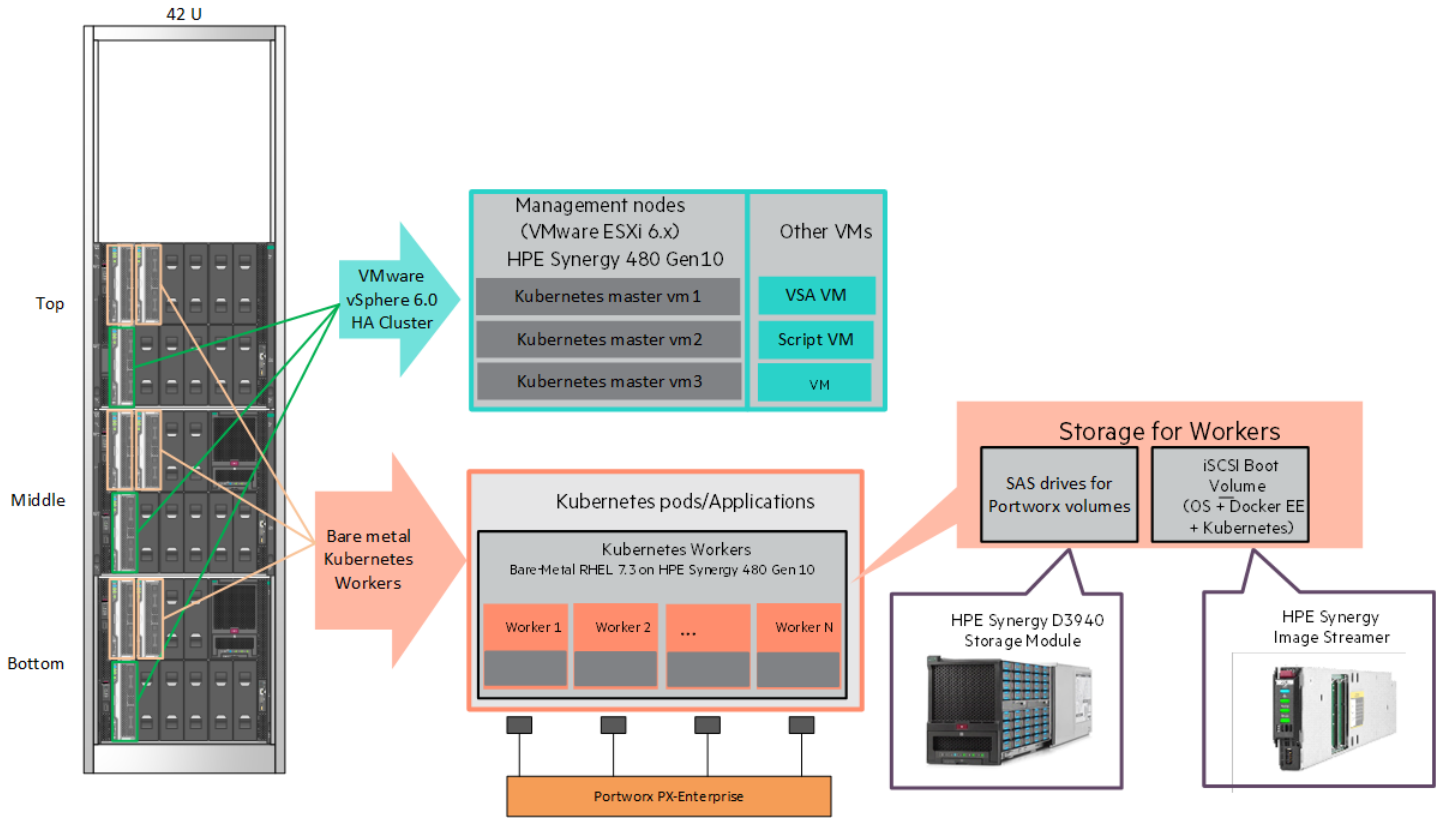

Figure 10. HPE Synergy configuration

The basic configuration of HPE OneView, such as network configuration, enclosure group configuration, and OS deployment server configuration, is done by the end user. In our Reference Configuration diagram shown in Figure 10, the compute modules shown in bay 7 of each frame are VMware ESXi hosts playing the role of management nodes. These three ESXi hosts are clustered to host redundant management VMs. The VMware ESXi software was installed on the local disk of each Synergy 480 Gen10 Compute Module following a manual install process for this RC (optionally VMware ESXi can also be deployed using the Image Streamer). In our Reference Configuration, server profile for bay 7 middle frame and server profile for bay 7 bottom frame are configured with storage from the HPE Synergy D3940 Storage Modules. We have used HPE StoreVirtual VSA (Virtual Storage Array) software to create a shared VM datastore environment from the disks provided by the D3940 Storage Modules to each of the two ESXi nodes. Optionally, you can create a shared datastore for the VMs using an HPE 3PAR array or using your existing shared storage solution.

The rest of the computes modules (non ESXi nodes) in the frames act as Kubernetes worker nodes. These compute modules will have a basic server profile without OS deployment plan. The section Setting up Kubernetes worker nodes using Image Streamer describes more in detail about the configuration and deployment through OneView server profiles and Image Streamer capabilities.

Setting up the Kubernetes master nodes

Once the VMs are provisioned with Red Hat® Enterprise Linux® (RHEL) 7.3 operating system, an Ansible script, listed in Appendix F, bootstraps the Kubernetes HA master cluster by configuring the required Kubernetes services and networking, configuring etcd cluster, and installing and configuring the dashboard. In this section, we will talk about "how to bootstrap the Kubernetes cluster using Ansible playbooks". Today, setting up an on-premises Kubernetes HA cluster is complex and time consuming and it is a manual process to install and configure all components. In our Reference Configuration, we are addressing this concern by developing an automated bootstrapping process using Ansible playbooks.

Below are the high-level steps for bootstrapping a Kubernetes master node cluster (HA setup):

- Three master nodes will be configured with Kubernetes services and Docker engine
- Configuring HA etcd cluster
- Configuring load balancer
- Installing Dashboard

For more details on setting up a Kubernetes master node cluster, running Ansible playbooks and sample code, go to GitHub at: https://github.com/HewlettPackard/K8s-Portworx-Synergy

Figure 11 shows the high-level flow to bootstrap the Kubernetes master nodes. The end user will invoke the Ansible command and in the background, the script talks to three VMs (can be physical servers), installs all components, configures various services, and then configures a virtual cluster address using a load balancer, and also configures the dashboard with a virtual cluster address for the user to interface with the Kubernetes cluster.

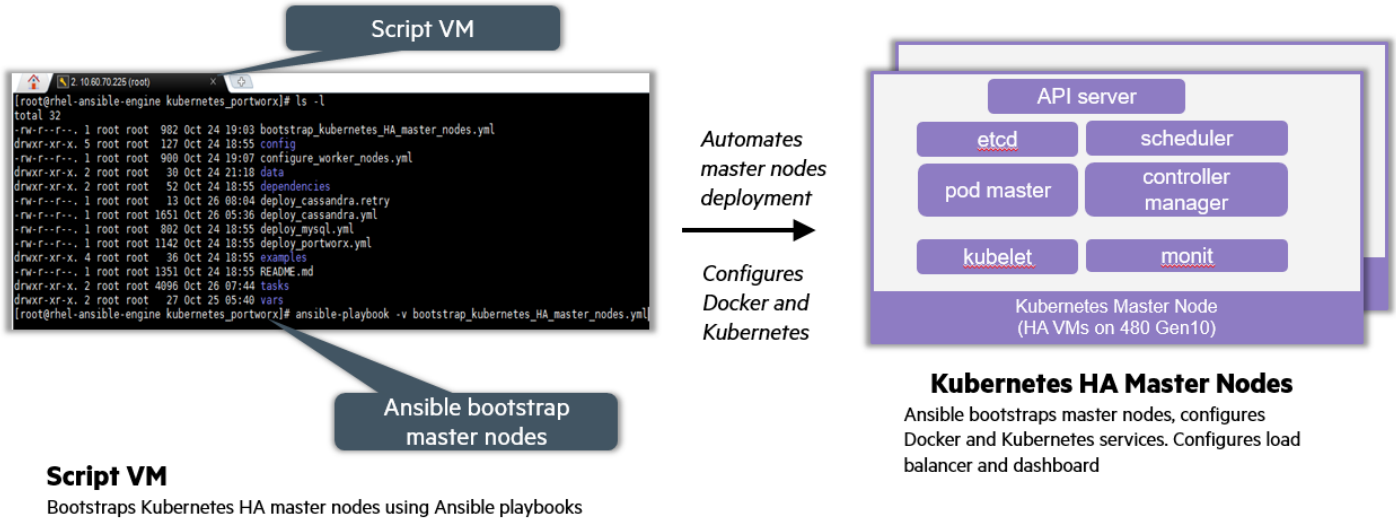

Figure 11. Bootstrap Kubernetes HA node

After bootstrapping is complete, the user can access the HA cluster using the cluster virtual IP or access the dashboard to operate the Kubernetes cluster and services.

Setting up Kubernetes worker nodes using Image Streamer

In this section we discuss Image Streamer capabilities that are used to provision bare metal Kubernetes worker nodes. In this process, the user should configure Image Streamer with plan scripts, OS golden images, and deployment plans. For more details on Image Streamer, refer to the HPE Image Streamer user guide for preparing Image Streamer from hpe.com/info/synergy-docs. Once Image Streamer is prepared and configured as an OS deployment server in OneView, the user can pre-populate server profiles on selected servers without an OS deployment plan. Optionally you can create a template-based approach to assign server profiles to selected servers. When there is a request from a user to provision Kubernetes worker nodes, the user updates the server profiles (or templates) with OS deployment plans to provision the Kubernetes worker nodes. Updating the server profiles speeds up the provisioning time instead of creating server profiles with OS deployment plans from scratch. In our Reference Configuration, instead of making changes in the OneView GUI, we are using Ansible playbooks to automate the Kubernetes worker node provisioning. Ansible scripts help in rapid deployment and bring agility to your business demands. The following describes how Image Streamer functions to provision the Kubernetes worker nodes.

Once the server profile is assigned to an available server, the OS deployment process begins through Image Streamer. A smart clone of the golden image is created as a new volume and presented to the server and the server is powered up. The server will automatically boot with the newly created OS volume. The build plan and plan scripts that are part of the RHEL-7.3-Kubernetes-Worker-2017-10-10 deployment plan will automatically customize the server with all the networking, storage connectivity, firewall, and other host settings including services related to Kubernetes and Docker engine. Figure 12 depicts how Image Streamer is utilized in our Reference Configuration.

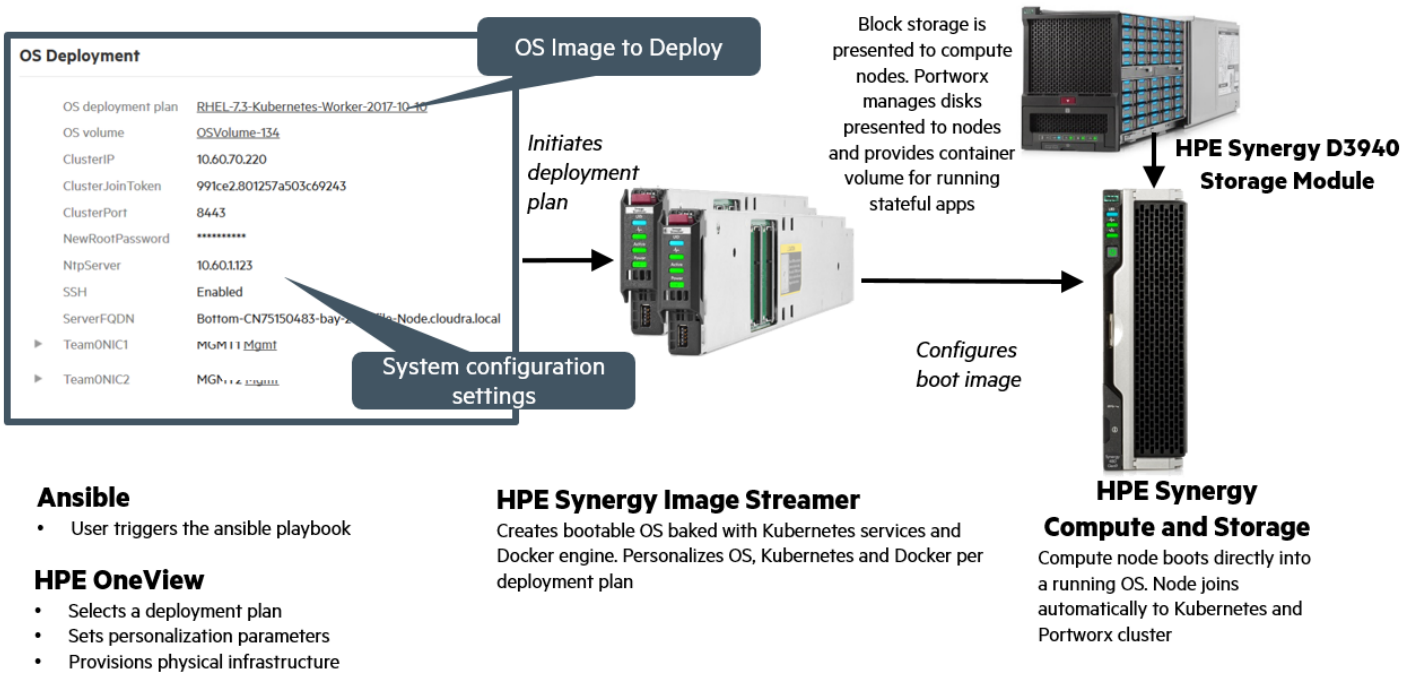

Figure 12. HPE Synergy Image Streamer configuration

The plan scripts also automate the completion of the worker configuration and joining the Kubernetes cluster. Once the server deployment is completed, the newly provisioned Synergy node is visible in the Kubernetes cluster as a new worker node and available to host new container workloads as shown in Figure 13.

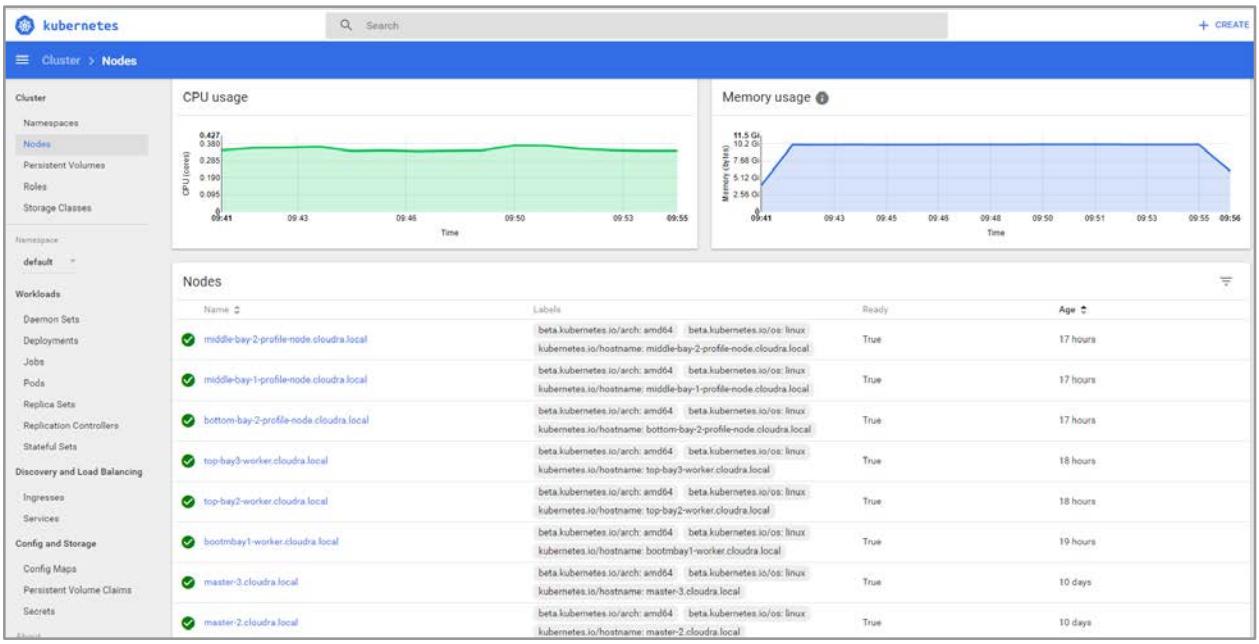

Figure 13. Kubernetes dashboard showing Synergy nodes

Setting up the Portworx software

In this section, we will talk about setting up the Portworx software on the Kubernetes cluster. In our Reference Configuration, Portworx software is deployed from the Kubernetes master node as a daemon set. Whenever a worker node joins the Kubernetes cluster, a Portworx component gets installed on that worker node automatically. We have automated the deployment of Portworx on Kubernetes using Ansible scripts. More details about Portworx deployment scripts can be found at the GitHub link listed in Appendix F. Optionally you can install Portworx on Kubernetes by following the instructions at https://docs.portworx.com/scheduler/kubernetes/install.html

Cassandra NoSQL workload in containers with Kubernetes and Portworx

This section covers Cassandra as a workload and deployment practices for Portworx and Kubernetes with HPE Synergy. Apache Cassandra, first developed at Facebook, is an open source distributed database management system designed to handle large amounts of data across scale-out servers. Cassandra has built-in data sharding and load-balancing with its clients.

This section provides the information to deploy Cassandra with Portworx and Kubernetes.

The benefits of containerizing Cassandra are:

- Increase the number of Cassandra application rings deployed per server including co-running with other stateful applications
- Easy scaled out, upgrade, and manage through modern schedulers like Kubernetes
- Portable across cloud-native environments through containerized compute and storage

The challenge with containerized Cassandra is that the storage needs to be isolated down to the container level; otherwise, Cassandra will undergo constant repair and degradation as schedulers move the compute instances. Through coupling a cloud-native storage product like Portworx with Kubernetes, Cassandra applications will benefit from increased density, faster recovery times during failures, and simplified deployments.

Deploying Cassandra

A Kubernetes service will be used as the stable endpoint for Cassandra clients. This service provides an IP address to which clients bind. Cassandra clients that call this service IP address will then be routed to Cassandra server containers. Those server containers will be deployed in pods that will be managed at the compute level by a Kubernetes StatefulSet.

The complete set of steps for deploying and verifying Cassandra are available in Appendix E: Complete steps with Kubernetes, Cassandra, and Portworx.

Cassandra shards its data amongst its server side containers. To take advantage of sharding[4], you will specify in the Cassandra Stateful spec to deploy with three pod instances. In your environment, you can deploy with dozens of instances. The pod spec is itself a section within a StatefulSet YAML. The StatefulSet manages the pods and specifies the container resource usage.

Expanding Kubernetes and failover Cassandra

Once a Cassandra StatefulSet and service has been created, you can write data into Cassandra, expand the Cassandra deployment to more worker nodes, and also expand the number of worker nodes. Because Portworx was deployed with Kubernetes as a daemon set, Portworx will automatically scale out whenever a new Kubernetes worker is added to the cluster. Hence, every server running Kubernetes can also support container-level storage.

For detailed steps on running with Cassandra, see Appendix E: Complete steps with Kubernetes, Cassandra, and Portworx.

Benefits of orchestration and automation

In the world of DevOps practice and microservices architecture, the infrastructure should have the agility and speed to respond to business demands. As described under section Solution setup, a Kubernetes cluster is bootstrapped using Ansible by combining HPE Synergy composable infrastructure capabilities through HPE OneView Ansible library for bare metal provisioning as Kubernetes worker nodes. Also configuration of Kubernetes services and installing Portworx is automated using Ansible.

Based on our testing in the lab, we observed the following deployment benefits of rapid deployment through automation versus manual installation and configuration

- About 15 minutes to set up Kubernetes HA master nodes versus 3 hours with the manual process. The manual process requires that the user is well versed with Kubernetes deployment.
- About 6 minutes to deploy HPE Synergy compute modules and join the Kubernetes cluster versus 30 minutes with the manual process. The manual process requires that the user is well versed with Kubernetes deployment, and OneView and Image Streamer operations.

Summary

Customers who want to run container workloads at scale and with a fully managed experience should consider HPE Synergy as the infrastructure with Kubernetes and Portworx as the container platform. This document described how enterprises can start with bare metal systems, rapidly configure physical resources, and build an entire container platform for hosting stateful microservices. HPE Synergy Image Streamer was used to rapidly deploy Kubernetes software on worker nodes, this enables automated scale-out capabilities based on application demands.

This Reference Configuration provided architectural guidance for deploying, scaling, and managing stateful workloads with Kubernetes and Portworx PX-Enterprise edition on HPE Synergy composable infrastructure. More specifically, it provided the following benefits:

- Offer Container-as-a-Service (CaaS) and managed container functionality within IT organizations using Kubernetes as the orchestration engine
- Run any stateful workloads using Portworx integrations with Kubernetes and Portworx PX-Enterprise for elastic container storage and container data lifecycle management
- Automate deployment and scale of an integrated Portworx + Kubernetes platform leveraging HPE Synergy strengths in rapid provisioning using HPE Synergy Image Streamer. Testing showed that with HPE Synergy Composer plus HPE Synergy Image Streamer we were able to

[4] Database Sharding can be simply defined as a "shared-nothing" partitioning scheme for large databases across a number of servers, enabling new levels of database performance and scalability achievable. (source: agildata.com/database-sharding)

deploy a new CaaS environment (Kubernetes + Portworx) in around 20 minutes. This time didn't include basic setting up of HPE Synergy Composer, HPE Synergy Image Streamer, and plain vanilla RHEL VMs.

Appendix A: Bill of materials

Note

Part numbers are at time of testing 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 5. Hardware BOM

Qty	Part Number	Description
		Rack and Network Infrastructure
1	BW908A	HPE 42U 600x1200mm Enterprise Shock Rack
1	BW909A	HPE 42U 1200mm Side Panel Kit
1	H8B49A	HPE Mtrd Swtchd 3.6kVA/60320/WW PDU
1	BW932A	HPE 600mm Rack Stabilizer 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
		HPE Synergy Frame 1 (Bottom)
1	797740-B21	HPE Synergy12000 CTO Frame 1xFLM 10x Fan
1	835386-B21	HPE Synergy D3940 CTO Storage Module
10	872475-B21	HPE 300GB SAS 10K SFF SC DS HDD
1	757323-B21	HPE Synergy D3940 IO Adapter
3	871942-B21	HPE SY 480 Gen10 CTO Prem Cmpt Mdl
3	872112-L21	HPE SY 480 Gen10 Xeon-S 4114 FIO Kit
3	872112-B21	HPE SY 480 Gen10 Xeon-S 4114 Kit
24	815100-B21	HPE 32GB 2Rx4 PC4-2666V-R Smart Kit
6	870753-B21	HPE 300GB SAS 15K SFF SC DS HDD
3	875242-B21	HPE 96W Smart Storage Battery 260mm Cbl
3	871573-B21	HPE SAS Cable for P416ie-m SR G10 Ctrlr
3	777430-B21	HPE Synergy 3820C 10/20Gb CAN
3	804428-B21	HPE Smart Array P416ie-m SR Gen10 Ctrlr
2	755985-B21	HPE Synergy 12Gb SAS Connection Module
1	779218-B21	HPE Synergy 20Gb Interconnect Link Mod
1	794502-B23	HPE VC SE 40Gb F8 Module
1	838327-B21	HPE SY Dual 10GBASE-T QSFP 30m RJ45 XCVR
4	720193-B21	HPE BLc QSFP+ to SFP+ Adapter
1	798096-B21	HPE Synergy 12000F 6x 2650W AC Ti FIO PS
1	804353-B21	HPE Synergy Composer

Qty	Part Number	Description
1	804938-B21	HPE Synergy 12000 Frame Rack Rail Option
1	804942-B21	HPE Synergy Frame Link Module
1	804943-B21	HPE Synergy 12000 Frame 4x Lift Handle
1	859493-B21	Synergy Multi Frame Master1 FIO
8	804101-B21	HPE Synergy Interconnect Link 3m AOC
2	720199-B21	HPE BLc 40G QSFP+ QSFP+ 3m DAC Cable
6	861412-B21	HPE CAT6A 4ft Cbl
2	720199-B21	HPE BLc 40G QSFP+ QSFP+ 3m DAC Cable
8	804101-B21	HPE Synergy Interconnect Link 3m AOC
9	861412-B21	HPE CAT6A 4ft Cbl
		HPE Synergy Frame 2 (Middle)
1	797740-B21	HPE Synergy12000 CTO Frame 1xFLM 10x Fan
1	835386-B21	HPE Synergy D3940 CTO Storage Module
10	872475-B21	HPE 300GB SAS 10K SFF SC DS HDD
1	757323-B21	HPE Synergy D3940 IO Adapter
3	871942-B21	HPE SY 480 Gen10 CTO Prem Cmpt Mdl
3	872112-L21	HPE SY 480 Gen10 Xeon-S 4114 FIO Kit
3	872112-B21	HPE SY 480 Gen10 Xeon-S 4114 Kit
24	815100-B21	HPE 32GB 2Rx4 PC4-2666V-R Smart Kit
6	870753-B21	HPE 300GB SAS 15K SFF SC DS HDD
3	875242-B21	HPE 96W Smart Storage Battery 260mm Cbl
3	871573-B21	HPE SAS Cable for P416ie-m SR G10 Ctrlr
3	777430-B21	HPE Synergy 3820C 10/20Gb CAN
3	804428-B21	HPE Smart Array P416ie-m SR Gen10 Ctrlr
2	755985-B21	HPE Synergy 12Gb SAS Connection Module
1	779218-B21	HPE Synergy 20Gb Interconnect Link Mod
1	794502-B23	HPE VC SE 40Gb F8 Module
1	838327-B21	HPE SY Dual 10GBASE-T QSFP 30m RJ45 XCVR
4	720193-B21	HPE BLc QSFP+ to SFP+ Adapter
1	798096-B21	HPE Synergy 12000F 6x 2650W AC Ti FIO PS
1	804353-B21	HPE Synergy Composer
1	804937-B21	HPE Synergy Image Streamer
1	804938-B21	HPE Synergy 12000 Frame Rack Rail Option
1	804942-B21	HPE Synergy Frame Link Module
1	859494-B22	Synergy Multi Frame Master2 FIO
		HPE Synergy Frame 3 (Top)
1	797740-B21	HPE Synergy12000 CTO Frame 1xFLM 10x Fan
3	871942-B21	HPE SY 480 Gen10 CTO Prem Cmpt Mdl
3	872112-L21	HPE SY 480 Gen10 Xeon-S 4114 FIO Kit
3	872112-B21	HPE SY 480 Gen10 Xeon-S 4114 Kit
24	815100-B21	HPE 32GB 2Rx4 PC4-2666V-R Smart Kit
6	870753-B21	HPE 300GB SAS 15K SFF SC DS HDD

Qty	Part Number	Description
3	875242-B21	HPE 96W Smart Storage Battery 260mm Cbl
3	777430-B21	HPE Synergy 3820C 10/20Gb CAN
2	779218-B21	HPE Synergy 20Gb Interconnect Link Mod
1	798096-B21	HPE Synergy 12000F 6x 2650W AC Ti FIO PS
1	804937-B21	HPE Synergy Image Streamer
1	804938-B21	HPE Synergy 12000 Frame Rack Rail Option
1	804942-B21	HPE Synergy Frame Link Module
1	861413-B21	HPE CAT6 10Ft Cbl

Table 6. Software BOM

Qty	Part number	Description
3	BD715AAE	VMware vSphere EntPlus 1P 3yr E-LTU
6	Q0K93AAE	HPE Docker Ent Std 1yr 24x7 E-LTU
6		Porxtworx PX-Enterprise Metal 1 yr 24x7

Appendix B: Deploying Kubernetes masters as virtual machines

In this section we provide a high-level overview of the Kubernetes master nodes cluster installation process, load balancer, etcd cluster, and dashboard. For specific installation and configuration processes, refer to the official Kubernetes documentation[5]. In this Reference Configuration, Kubernetes master nodes are deployed as VMware virtual machines. To deploy a high-availability (HA) Kubernetes cluster, a minimum of three virtual machines are needed. It is assumed that three RHEL 7.3 VMs are installed before deploying the Kubernetes master nodes through Ansible scripts listed in Appendix F. Typically setting up clustered Kubernetes master nodes on-premises is complex and time consuming. To address this concern, we have developed Ansible playbooks to deploy three master nodes. Based on our internal testing an HA Kubernetes cluster may be deployed in 15 minutes including configuration of external load balancer, external etcd cluster, and dashboard. Refer to the Ansible playbooks listed in Appendix F for more technical details.

Once the Kubernetes master nodes cluster is ready, the Ansible playbook orchestrates the provisioning of worker nodes through Synergy Composer and Image Streamer REST APIs. We leverage the OneView Ansible library to provision the physical servers. Using Image Streamer customization plan scripts, Synergy worker nodes are automatically joined to the Kubernetes master cluster. This feature is helpful to address the concern of growing the Kubernetes cluster on demand based on workload requirements and minimizing the manual user interaction to build the new worker node and add to the cluster.

```
$ ansible-playbook -v update_multiple_server_profile_with_storage.yml

$ ansible-playbook -v update_multiple_server_profile_without_storage.yml
```

[5] https://kubernetes.io/docs/home/

Appendix C: Kubernetes and Portworx worker node plan custom attributes

In this section we talk about the Image Streamer custom attributes related to Kubernetes worker node. The OS golden image is backed with Kubernetes related services, such as docker, kubelet, and kubeadm, and it is being managed and customized through Image Streamer. When a server is selected to be updated with an OS deployment plan, an end user or a script will input custom attributes, such as the kubernetes cluster IP address, port, token and other host specific details; so that when the node is provisioned, all Kubernetes related services are up and running and the node will be automatically joined to the master cluster. Portworx is installed and deployed once as a daemon set through the kubectl command on the Kubernetes master cluster nodes. There are no Portworx components baked into the OS golden image. Figure 14 shows the Kubernetes related custom attributes.

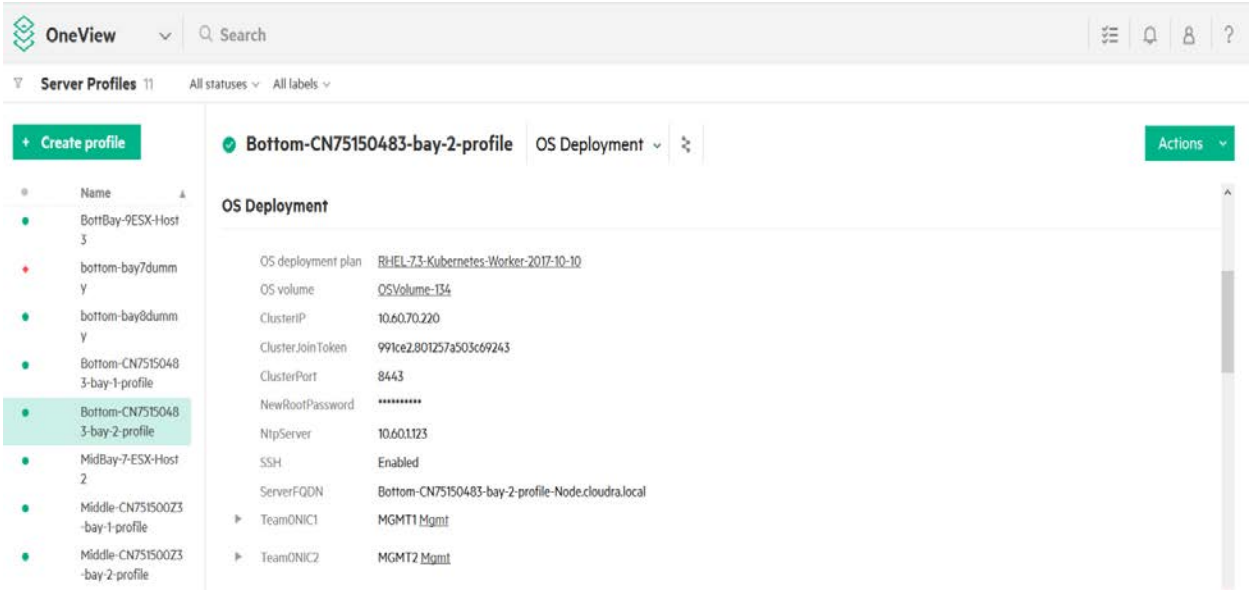

Figure 14. Image Streamer customer attributes passed to OS deployment plan

List of custom attributes (Ansible variables)

The following table lists custom attributes (Ansible variables) from the Ansible yaml file. The complete yaml file is at the GitHub link mentioned in Appendix F.

Table 7. Sample custom attributes (Ansible variables) present in update_multiple_server_profile_without_storage.yml

```
- connection: local
  hosts: localhost
  vars:
    config: '{{ playbook_dir }}/oneview_config.json'
    profile_numbers: 4
    scope_name: "kubeworkergroup1"
#network variables
    management_network_name: "Mgmt"
    deployment_network_name: "Deploy"
#OS deployment plan variables
    os_deployment_plan_name: "RHEL-7.3-Kubernetes-Worker-2017-10-10"
    cluster_ip: "{{ lookup('ini', 'CLUSTER_IP section=cluster_data file={{ playbook_dir
}}/../kubernetes_portworx/data/cluster_data.ini') }}"
    cluster_join_token: "{{ lookup('ini', 'CLUSTER_TOKEN section=cluster_data file={{ playbook_dir
}}/../kubernetes_portworx/data/cluster_data.ini') }}"
    cluster_port: "{{ lookup('ini', 'CLUSTER_PORT section=cluster_data file={{ playbook_dir
}}/../kubernetes_portworx/data/cluster_data.ini') }}"
    mgmt_NIC1_connection_id: "1"
    mgmt_NIC1_dhcp: "false"
    mgmt_NIC1_ipv4disable: "false"
    mgmt_NIC1_networkuri: "/rest/ethernet-networks/ce53067d-42c6-4603-aba7-f53df8c2f78c"
    mgmt_NIC1_constraint: "auto"
    mgmt_NIC2_connection_id: "2"
    mgmt_NIC2_dhcp: "true"
    mgmt_NIC2_ipv4disable: "false"
    mgmt_NIC2_networkuri: "/rest/ethernet-networks/ce53067d-42c6-4603-aba7-f53df8c2f78c"
    mgmt_NIC2_constraint: "dhcp"
    nic_teaming: "Yes"
    ntp_server: "10.60.1.123"
    root_password: "password"
    SSH: "Enabled"
```

Appendix D: Monitoring Portworx Volumes

As a best practice in production, monitoring should be configured to provide a unified view of metrics for container compute and storage. Figure 15 shows Grafana charts that plot container-level metrics for storage I/O, storage, CPU, and memory. To continue the example with Cassandra, Figure 7 shows the Cassandra volume based on its label. This label is the same as what is shown in Figure 15. An IT operator can quickly probe metrics and correlate against the volumes capacity.

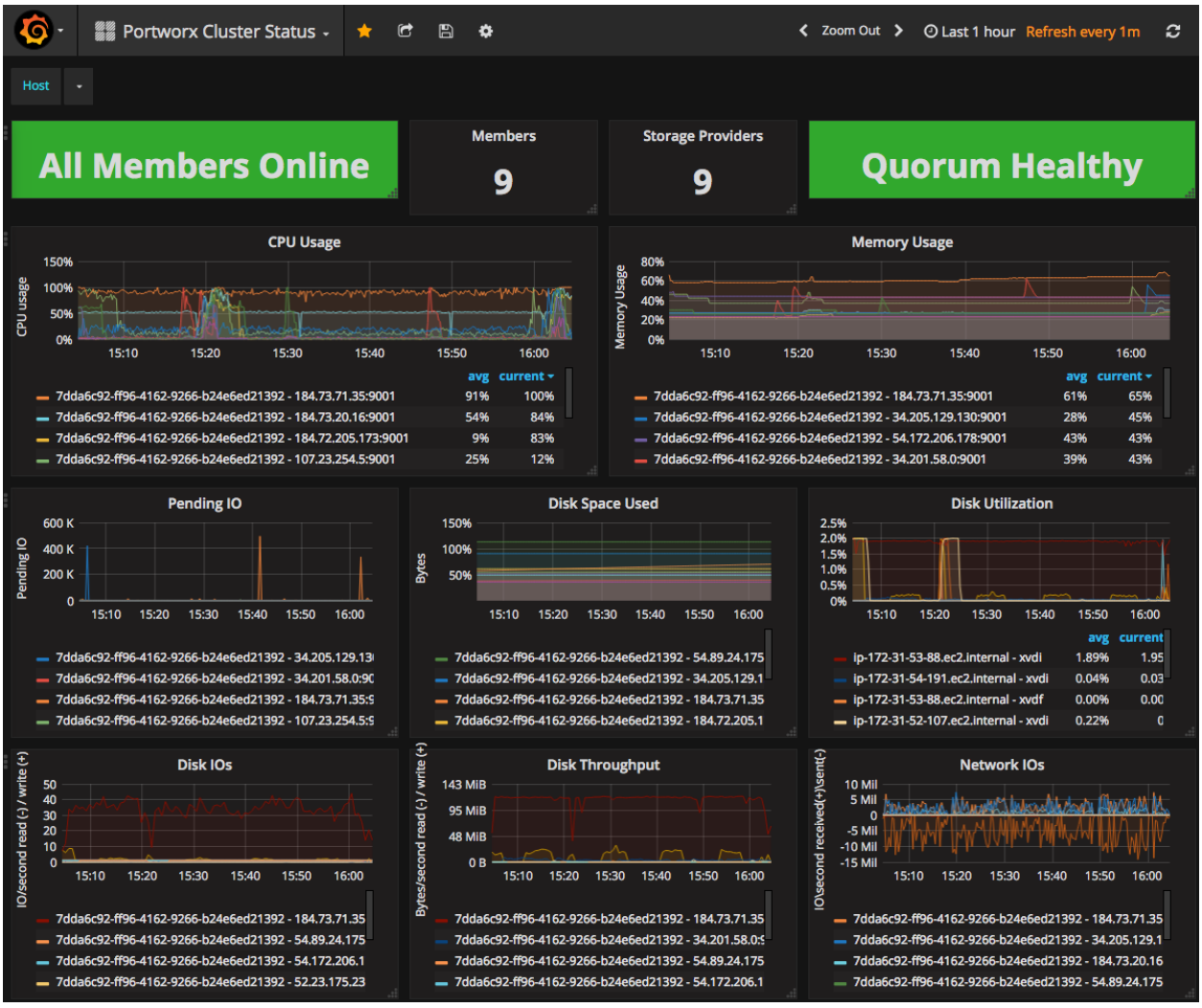

Figure 15. Grafana: Monitoring server nodes compute and storage

Appendix E: Complete steps with Kubernetes, Cassandra, and Portworx

In continuation from the section Cassandra NoSQL workload in containers with Kubernetes and Portworx, the following completes the steps to deploy and run Cassandra.

First, write the following *cassandra-headless-service.yml* in a file and save in the local directory:

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: cassandra
  name: cassandra
spec:
  clusterIP: None
  ports:
    - port: 9042
  selector:
    app: Cassandra
```

Apply the configuration:

```
$ kubectl apply -f cassandra-headless-service.yml
```

We do not need to create Portworx volumes for Cassandra as the volumes will be dynamically created when the pods are instantiated. However, we can specify the quality of storage to be consumed by volumes through one or more StorageClasses. The StorageClass below specifies that the Portworx volume should have block-level replication with two replicas and on the fastest tier of storage. (For more on Portworx volumes, see Dynamic Provisioning with Portworx Volumes.)

Create a px-storageclass.yml with the following:

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: px-storageclass
provisioner: kubernetes.io/portworx-volume
parameters:
  repl: "2"
  priority_io: "high"
```

Apply the configuration:

```
$ kubectl apply -f px-storageclass.yml
```

Create a *cassandra-statefulset.yml* with the following:

```
apiVersion: "apps/v1beta1"
kind: StatefulSet
metadata:
  name: cassandra
spec:
  serviceName: cassandra
```

```
  replicas: 3
  template:
    metadata:
      labels:
        app: cassandra
    spec:
      containers:
      - name: cassandra
        image: gcr.io/google-samples/cassandra:v11
        imagePullPolicy: Always
        ports:
        - containerPort: 7000
          name: intra-node
        - containerPort: 7001
          name: tls-intra-node
        - containerPort: 7199
          name: jmx
        - containerPort: 9042
          name: cql
        resources:
          limits:
            cpu: "500m"
          requests:
           cpu: "500m"
        securityContext:
          capabilities:
            add:
              - IPC_LOCK
        lifecycle:
          preStop:
            exec:
              command: ["/bin/sh", "-c", "PID=$(pidof java) && kill $PID && while
ps -p $PID > /dev/null; do sleep 1; done"]
        env:
          - name: MAX_HEAP_SIZE
            value: 512M
          - name: HEAP_NEWSIZE
            value: 100M
          - name: CASSANDRA_SEEDS
            value: "cassandra-0.cassandra.default.svc.cluster.local"
          - name: CASSANDRA_CLUSTER_NAME
            value: "K8Demo"
          - name: CASSANDRA_DC
            value: "DC1-K8Demo"
          - name: CASSANDRA_RACK
            value: "Rack1-K8Demo"
          - name: CASSANDRA_AUTO_BOOTSTRAP
            value: "false"
          - name: POD_IP
            valueFrom:
              fieldRef:
                fieldPath: status.podIP
          - name: POD_NAMESPACE
```

```
            valueFrom:
              fieldRef:
                fieldPath: metadata.namespace
        readinessProbe:
          exec:
            command:
            - /bin/bash
            - -c
            - /ready-probe.sh
          initialDelaySeconds: 15
          timeoutSeconds: 5
        # The volumes mounts below are persistent.
        # Each Cassandra replica gets its own volume mount.
        volumeMounts:
        - name: cassandra-data
          mountPath: /cassandra_data
  # These are converted to volume claims by the k8s controller
  # and mounted at the paths mentioned above.
  volumeClaimTemplates:
  - metadata:
      name: cassandra-data
      annotations:
        volume.beta.kubernetes.io/storage-class: px-storageclass
    spec:
      accessModes: [ "ReadWriteOnce" ]
      resources:
        requests:
          storage: 1Gi
```

To see the pods and whether each pod is scheduled, run the following:

```
$ kubectl get pods -l app=cassandra -o json | jq '.items[] | {"name": .metadata.name,"hostname": .spec.nodeName,
"hostIP": .status.hostIP, "PodIP": .status.podIP}'
```

Result from our setup shows:

```
{
  "name": "cassandra-0",
  "hostname": "k8s-2",
  "hostIP": "10.142.0.5",
  "PodIP": "10.0.160.2"
}
{
  "name": "cassandra-1",
  "hostname": "k8s-0",
  "hostIP": "10.142.0.3",
  "PodIP": "10.0.64.2"
}
{
  "name": "cassandra-2",
  "hostname": "k8s-1",
  "hostIP": "10.142.0.4",
  "PodIP": "10.0.192.3"
```

```
}
```

Appendix F

The Ansible playbooks developed for automated solution deployment is published at https://github.com/HewlettPackard/K8s-Portworx-Synergy

Resources and additional links

HPE Reference Architectures
hpe.com/info/ra

HPE Servers
hpe.com/servers

HPE Storage
hpe.com/storage

HPE Networking
hpe.com/networking

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

HPE Synergy
hpe.com/synergy

HPE and Docker Solutions
http://h22168.www2.hpe.com/us/en/partners/docker

Kubernetes
https://kubernetes.io

Portworx
https://portworx.com

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

Sign up for updates

a00039700enw, January 2018

Hewlett Packard Enterprise