Reference Architecture: Red Hat OpenShift Container Platform on Lenovo ThinkSystem Servers

Last update: 24 April 2020
Version 2.0

Provides overview of application containers using Lenovo ThinkSystem Servers

Describes container orchestration technologies including Docker and Kubernetes

Describes DevOps and Continuous Integration and Continuous Delivery

Provides examples for typical and Intel Select configurations

Xiaotong Jiang
Jintao Xu
Xifa Chen
Weixu Yang
Srihari Angaluri
Mike Perks
Billzheng Sun

LENOVO PRESS

Table of Contents

1 Introduction

The target audience for this Reference Architecture (RA) is system administrators or system architects. Some experience with Docker and OpenShift technologies may be helpful, but it is not required.

Emerging software applications are making use of containerization to enable rapid prototyping, testing, as well deployment to the cloud. The micro-service revolution introduced container-based virtualization, which offers many benefits when compared to traditional virtualization technologies. Containers provide a more portable and faster way to deploy services on cloud infrastructures compared to virtualization.

While containers themselves provide many benefits, they are not easily manageable in large environments. Hence, many container orchestration tools have increased in momentum and gained popularity. Each orchestration tool is different; hence they should be chosen individually for specific purposes. The Red Hat OpenShift® Container Platform uses Kubernetes which is an orchestration framework based on container-deployment practices. Kubernetes has gained popularity in the cloud community due to its maturity, scalability, performance, and many built-in tools that enable production-level container workload orchestration.

Red Hat OpenShift Container Platform 4.2 by Red Hat is built around a core of application containers powered by CRI-O, with orchestration and management provided by Kubernetes, on a foundation of Red Hat® Enterprise Linux (RHEL) or Red Hat Enterprise Linux CoreOS (RHCOS). It provides many enterprise-ready features like enhanced security, multitenancy, simplified application deployment, and continuous integration/continuous deployment tools. With Lenovo™ servers and Lenovo Open Cloud – Automation (LOC-A) technologies, deployment, provisioning and managing the Red Hat OpenShift Container Platform infrastructure becomes effortless and produces a resilient solution.

This RA describes the system architecture for the Red Hat OpenShift Container Platform 4.2 based on Lenovo ThinkSystem servers and Lenovo network switches. It provides detail of the hardware requirements to support various OpenShift node roles and the corresponding configuration of the systems. It also describes the network architecture and details for the switch configurations. The hardware bill of materials is provided for all required components to build the OpenShift cluster.

An example deployment configuration is described for a typical configuration. The hardware bill of materials is provided for all required components to build the OpenShift cluster. A deployment guide is provided to show how to prepare, provision, deploy, and manage the Red Hat OpenShift Container Platform on Lenovo ThinkSystem servers and Lenovo network switches.

2 Business problem and business value

2.1 Business problem

Businesses today want to deliver new features and updates to their products for their internal users as well as external stakeholders quickly and with high quality. Every industry today is seeing a transformation, which is predominantly driven by advances in technology. In order to stay competitive and relevant in their respective industry and marketplace, every business needs to take advantage of new technologies quickly and adopt them to their products and solutions. Today, much of the technology advancement and innovation is driven through a combination of software and hardware. More importantly, emerging technologies such as artificial intelligence (AI) and machine learning (ML) are fuelled by rapid advancements in software. In addition, many of the IT infrastructure and data center advancements are driven through software defined technologies such as software defined storage (SDS) and software defined networking (SDN). Hence software is a key driver in pushing forward various technologies in all industries.

Container technology has picked up momentum in the software development area and enabled developers to take advantage of several benefits from packaging their applications as containers:

- Containers are light-weight application run-time environments compared to virtual machines and are therefore less resource intensive and highly efficient.
- Containers enable developers to package their applications as well as all the library dependencies to properly run them so that a container image provides a completely self-sufficient environment to execute the application code. This also means that multiple application instances requiring different versions of the same libraries can be packaged into different containers and run side-by-side on the same operating system instance without any interference.
- Containers are portable across different platforms (as long as the underlying operating systems are compatible). Docker is a well-known open source project that provides the run-time abstraction and facilities to build and run containerized applications in a portable fashion.
- Containers are now the de facto standard of operation for some of the well-known public cloud environments including Google, Amazon AWS, and Microsoft Azure. Hence, applications packaged as containers can be executed on-prem or on public cloud without any modifications (other than additional steps to security the software for use on the public cloud).
- There are many open source tools available now to help developers easily create, test, and deploy containerized applications. In addition, all the well-known protocols for security, authentication, /authorization, storage, etc., can be applied to containerized workloads without any modifications to applications. In other words, you can take a legacy application written in a language such as Java and package it, as is, into a container image and run it.
- Containers are now the way to implement a continuous integration/continuous delivery (CI/CD) development pipeline and the DevOps paradigm of combining software development and infrastructure operations.

2.2 Business value

Software development life cycle (SDLC) practices have evolved to achieve high velocity and efficiency of development. Organizations today implement Agile/Scrum as the primary methodology to create cohesive development teams that work close to their customers, gather incremental product requirements, and deliver new features in short development cycles.

2.2.1 DevOps Overview

DevOps is evolving as a standard practice in many organizations to bring together software development and IT operations teams for the goal of eliminating process bottlenecks in development, quality assurance (QA), and delivery cycles, and servicing their end customers efficiently. Implementing a proper DevOps process requires careful planning and an assessment of the end-to-end pipeline from development to QA to delivery. Automation is a key aspect of DevOps. Traditional software development processes were handled mostly manually. When developers commit code to the source code repository, the test engineer or a build engineer would then checkout the code and build the project, resolving any conflicts. After the QA iterations, the release engineer would be responsible to take the release branch code and build the final shippable product. Along this pipeline, many of the steps were handled manually by people, which introduced the delays in the release cycles. Agile development now takes advantage of new automation tools that remove the manual steps.

Another core aspect of DevOps is providing the necessary freedom and resources to develop and test code without having to rely on IT operations teams to re-provision or re-configure hardware every time. With the advances in technologies including virtualization, containers, Cloud multi-tenancy, self-service, and so on, it is now possible to detach applications and end-users from physical hardware and provide the necessary tools for them to create the right virtual environment to run their applications without directly modifying the physical hardware or interfering with other users' applications. With cloud self-service, users can request and provision the hardware to meet their application specific requirements. Cloud administrators create the proper policy and authorization workflows such that the provisioning process does not require manual steps. DevOps essentially combines the role of the software engineer with that of the IT operator so that the end-to-end software pipeline can be implemented with automation.

2.2.2 Monolithic Vs Micro-services Architecture

Software architecture over the last two to three decades has evolved from a monolithic application that essentially delivered all feature functions in a single package to service-oriented architectures where the application is divided up into multiple tiers with each tier providing programming interfaces (APIs) for its clients to access the features via service calls. Software principles such as modularity, coupling, code reuse, etc., have remained the core principles that people still use, however new programming languages, runtime facilities, mobile versus cloud native techniques, etc., have evolved in the recent years to shift software from the traditional architectures to more of a micro-service architecture.

Micro-services are software applications that are organized around smaller subsets of functionality of the overall application such that they are much more manageable than a bulky piece of software developed by 10s of developers and coordinated in a complex dev/test process. Micro-services are software modules that run as services with open APIs. They use open protocols, e.g. HTTP, and expose REST based APIs so that the services can run anywhere - on-premises or on the public cloud, and still be locatable via well-defined

service end-points. Due to this design, micro-services provide a loosely-coupled architecture that can be maintained by smaller development teams and can be independently updated.

Containers provide a natural mechanism to implement micro-services because they allow you to package the application code and all its runtime library dependencies into a single image, which is portable across various platforms. In addition, container orchestration platforms such as Kubernetes provide the mechanisms for service location, routing, service replication, etc., which helps micro-services development and runtime. Developers do not need to explicitly write additional code for these types of services because the platform provides these facilities.

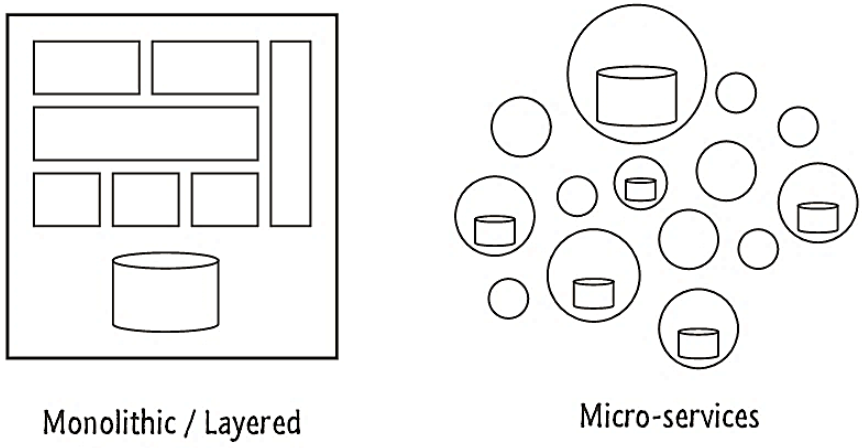

Figure 1. Moving from a monolithic to Micro-services architecture

2.2.3 Continuous Integration/Continuous Delivery (CI/CD)

As discussed in the previous section, successful DevOps practice requires a good amount of automation of the development, test, QA, and delivery pipeline. This is where the CI/CD comes into play.

Continuous integration is the process by which new code development through build, unit testing, QA, and delivery is automated end-to-end using build tools and process workflows. CI enables rapid integration of code being developed by multiple engineers concurrently and committed into a source code repository. CI enables rapid build and test of code so that software bugs and quality issues are identified quickly. Once the code passes the test plan and QA it can then be pushed to the release branches for release integration.

Continuous Delivery (CD) enables automation around delivering code to production systems after performing the necessary functional, quality, security, and performance tests. Continuous delivery enables bringing new features in the software to the end users faster without going through the manual release test and promotion steps.

More information on CI/CD with OpenShift is available in the following online book:
assets.openshift.com/hubfs/pdfs/DevOps_with_OpenShift.pdf

Figure 2 shows a high-level view of DevOps pipeline.

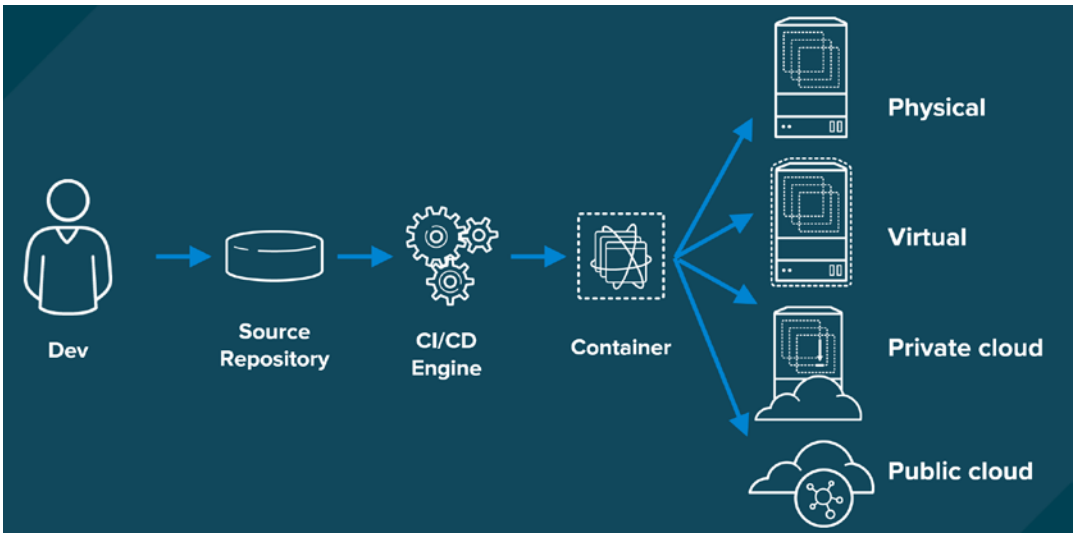

Figure 2. DevOps pipeline from a high-level

As described previously, source code from multiple concurrent developers is integrated, tested, and deployed to production through automation tools. The OpenShift Container Platform provides the mechanisms to implement the CI/CD pipelines with tools such as CloudBees Jenkins. See the following post on how to do this: blog.openshift.com/cicd-with-openshift/.

2.2.4 Developer Toolbox

Developers can use new tools, and plugins for container builds, CI/CD pipelines, and serverless deployments in a new application development-focused user interface. Developers will have a streamlined Kubernetes experience in developer portal. This empowers developers to focus on coding rather than dealing with the specifics of Kubernetes operations. See the following post for more details about the toolbox: openshift-developers-toolbox.

3 Requirements

The functional and non-functional requirements for this reference architecture are desribed below.

3.1 Functional requirements

Table 1 lists the functional requirements.

Table 1. Functional Requirements

Requirement	Description
Container orchestration services	Red Hat OpenShift Container Platform is designed to run workload container images at scale using the Kubernetes container orchestrator, and container runtime interface (CRI-O).
User self-service	OpenShift supports a Web based UI console that allows users to login and manage their containerized workloads.
Policy management	OpenShift allows administrators to configure role-based authorization to manage the system resources such as compute, networking, and storage, and application workloads.
Cloud integration	OpenShift supports an integrated container registry, the Quay container registry, or public registries which allow users to pull down container images from other places. In addition, building container images on OpenShift platform allows portability to other clouds such as Google container engine.
Network and Storage virtualization	Through built-in OpenShift networking and storage services for Kubernetes, users can access these abstracted resources through their container applications. In addition, OpenShift provides network infrastructure services through open protocols such as VXLAN. A variety of storage facilities can be exposed to container applications via the Kubernetes persistent volume plug-ins and stateful sets.
Command line tools	OpenShift container platform provides CLI tools for almost all cluster operations and for container image operations. In addition, administrators can use Kubernetes CLI tools to directly access its services.
CI/CD tools	A variety of open source and commercial tools are available such as Jenkins build server and integration with GitHub source code repository to implement CI/CD pipelines.
Open source tools	Red Hat container registry and other open container registries such as dockerhub are available to OpenShift users to access open source tools such as nginx, apache httpd, mysql, postgres, Cassandra, etc.
Automation tools	Many tools are available for automation including Ansible, Chef, Puppet, etc.
Service mesh	OpenShift Service Mesh is based on Istio along with the Kiali and Jaeger projects and delivered via Operator. OpenShift Service Mesh provides traffic monitoring,

	access control, discovery, security, resiliency, tracing, and reporting to a group of services by running as container sidecars.

3.2 Non-functional requirements

Table 2 lists the non-functional requirements that are needed for typical OpenShift deployments

Table 2. Non-functional Requirements

Requirement	Description
Scalability	The OpenShift Container Platform is designed for scale. The platform allows for hundreds of containerized workloads to be scheduled and run without any performance bottlenecks. The physical resources such as compute nodes and storage can be scaled as the workload and user base grows.
Load balancing	OpenShift master nodes provide the core API and management services for the Kubernetes cluster. For production environments, you can implement load-balancing and heartbeat monitoring for the core services via HAproxy and Keepalived. User can also use commercial LB such as F5/A10/etc for their production environment. In addition, Kubernetes handles load-balancing of the workload containers through built-in scheduler features, network routing, replication services, etc.
Fault tolerance	Fault tolerance can be provided to critical container workloads such as databases via Kubernetes built-in mechanisms. In addition, data and configuration settings for container images can be persisted across instances via persistent volume claims and stateful sets.
Physical footprint	OpenShift container platform can be implemented with as little as a single node where all services are consolidated and scaled through multiple physical nodes to distribute services and containers. Hence, the architecture is quite flexible and allows to start small and then scale out.
Ease of installation	Ansible playbooks are available to automate the OpenShift Container Platform deployment.
Ease of management/operations	Administrator tools and OpenShift Web console allow day 2 management operations to be performed. In addition, the Lenovo XClarity Administrator tool enables hardware monitoring and management.
Flexibility	OpenShift container platform can be deployed both in a development/test setting and production setting. Various options are available for third party network and storage implementation for OpenShift.
Security	Red Hat OpenShift Container Platform has built-in enterprise grade security, all the way from the operating system layer up to the container registries. Both built-in authentication/authorization facilities and external authentication/authorization integration with tools such as OpenLDAP are supported.

High performance	OpenShift and Kubernetes have achieved wide industry adoption due to the robustness of the platform and high-performance. Enterprises can implement very large-scale OpenShift environments to support hundreds of users and thousands of container workloads with no performance bottlenecks.

4 Architectural overview

The OpenShift Container Platform is a complete container application platform that provides all aspects of the application development process in one consistent solution across multiple infrastructure footprints. OpenShift integrates all of the architecture, processes, platforms, and services needed to help development and operations teams traverse traditional siloed structures and produce applications that help businesses succeed.

Figure 3 below shows the high level architecture of the Red Hat OpenShift Container Platform and the core building blocks. OpenShift is a platform designed to orchestrate containerized workloads across a cluster of nodes. The system uses the Kubernetes as the core container orchestration engine, which manages the Docker container images and their lifecycle.

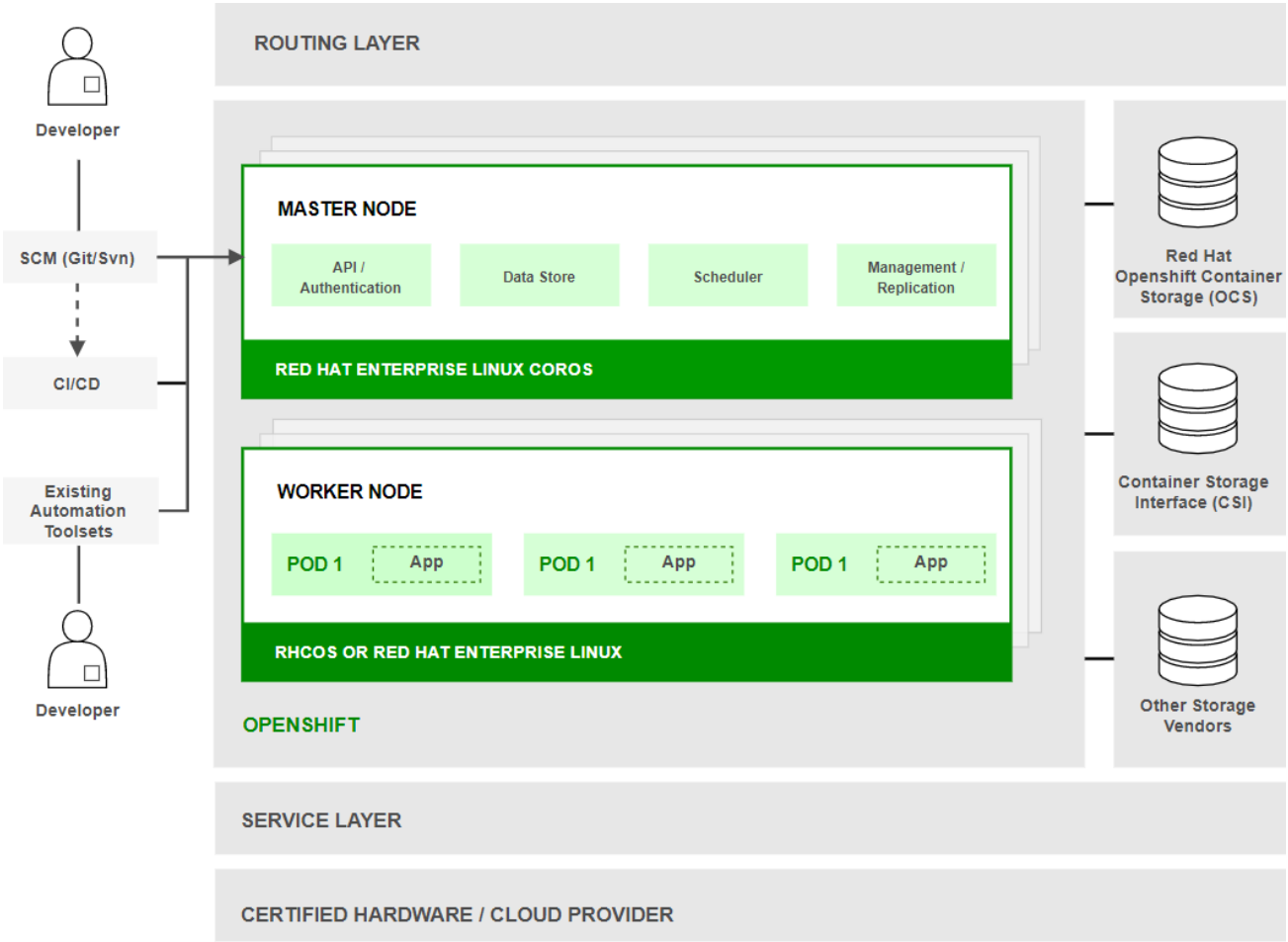

Figure 3. Red Hat OpenShift Container Platform Architecture

The physical configuration of the OpenShift platform is based on the Kubernetes cluster architecture. The master node is the primary node on which the Kubernetes scheduler, along with the distributed cluster data store (etcd), the REST API services, and other associated management services run. In a product environment, you need to ensure high availability of the master services through replicating the services to multiple physical servers and implementing monitoring and load-balancing services such as Keepalived and

HAproxy. Worker nodes run the users containerized applications on top of the CRI-O container runtime environment.

5 Component model

As shown in Figure 4, this chapter describes the components and logical architecture of the Red Hat OpenShift solution.

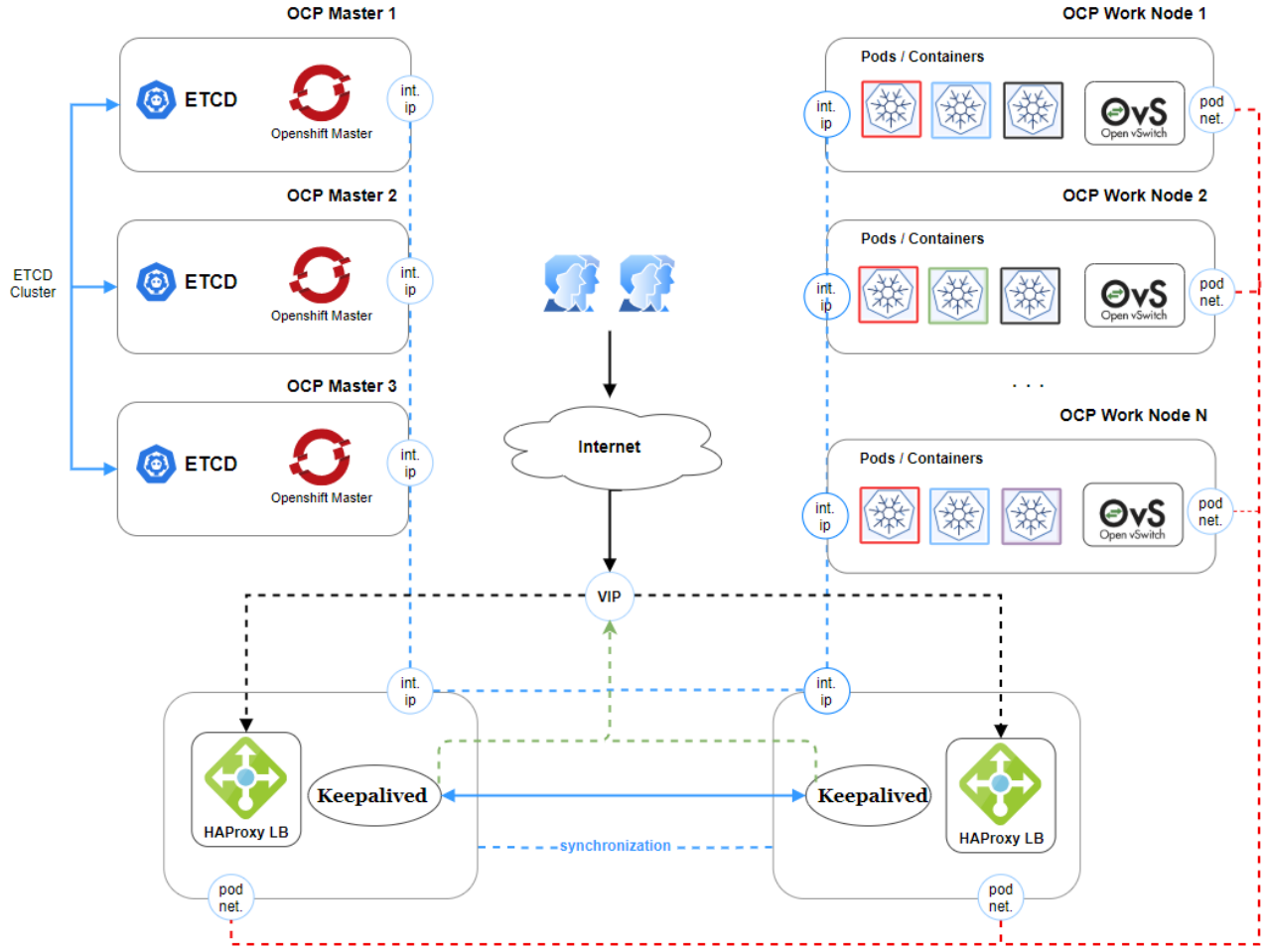

Figure 4. Red Hat OpenShift Container Platform logical architecture

All the OpenShift nodes are connected via the internal network, where they can communicate with each other. Furthermore, Open vSwitch creates its own network for OpenShift pod-to-pod communication. Because of the multi-tenant plugin, Open vSwitch pods can communicate to each other only if they share the same project namespace. There is a virtual IP address managed by *Keepalived* on two *load balance (LB)* hosts for external access to the OpenShift web console and applications. Applications can use local volume/host path, storages support CSI interface, OpenShift container storage (OCS), etc as backend storage in OpenShift cluster.

5.1 OpenShift infrastructure components

Figure 5 shows the four types of nodes in OpenShift cluster: *bastion, LB, master, and worker.* A temporary bootstrap node is not shown in Figure 5. Bootstrap node can be removed from cluster after deployment. Bootstrap node introduction is in section5.1.3. Backend Storage is not listed in this section; Storage introduction is in section 6.5.

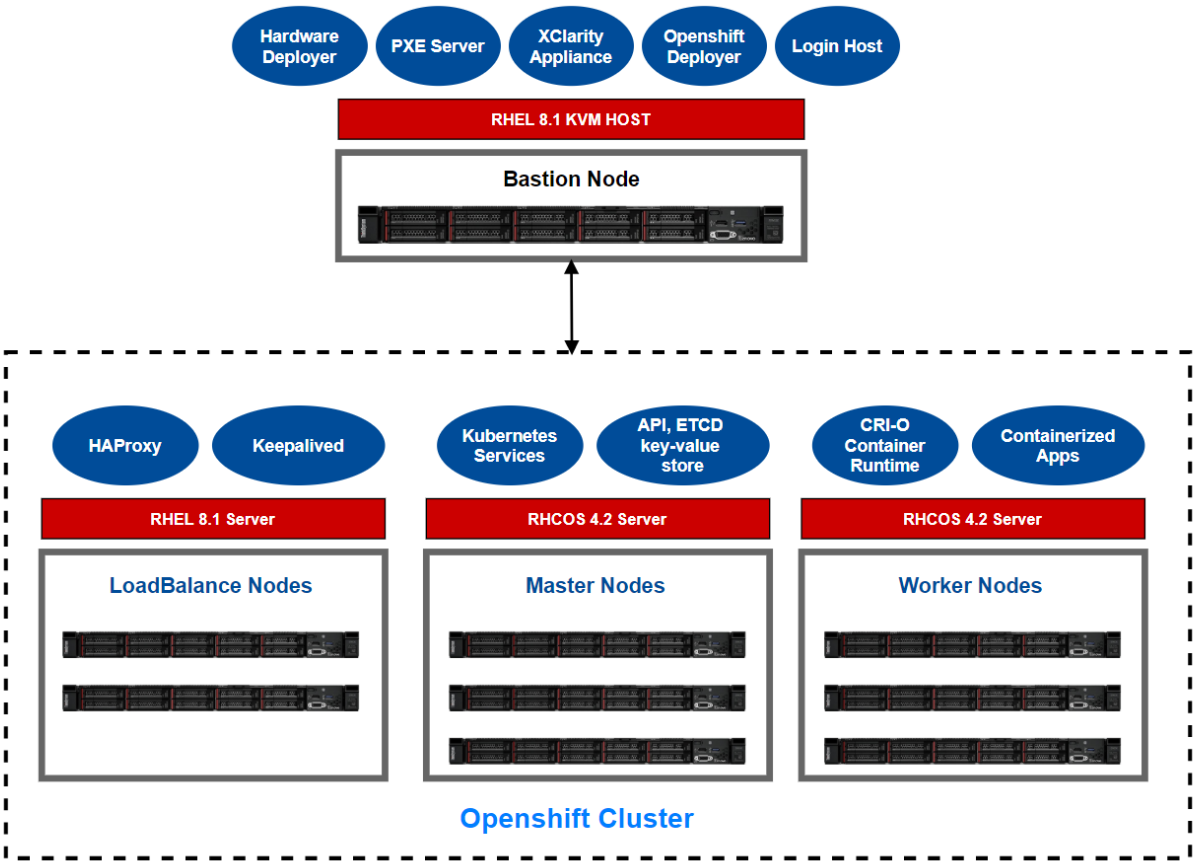

Figure 5. OpenShift Nodes

5.1.1 Bastion node

This is a dedicated node that serves as the main deployment and management server for the OpenShift cluster. This is used as the logon node for the cluster administrators to perform the system deployment and management operations. OpenShift installation file or Lenovo Open Cloud-Automation (LOC-A) tools is running on Bastion node for manual deployment or auto-deployment of OpenShift Container Platform. In addition, this node is also used for hardware management via tools such as xCAT and Lenovo XClarity Administrator. The *Bastion* node runs RHEL 8.1 or CentOS 8.1 Server with the Linux KVM packages installed.

5.1.2 Load Balance node

The OpenShift *Load Balance* node runs load balance services such as the Keepalived and the HAProxy router. The HAProxy router provides routing functions for OpenShift applications. It currently supports HTTP(S) traffic and TLS-enabled traffic via Server Name Indication (SNI). Additional applications and services can be deployed on OpenShift *Load balance* nodes. The OpenShift *Load balance* node runs RHEL Server 8.1 or CentOS 8.1.

5.1.3 Bootstrap node

OpenShift Container Platform uses a temporary bootstrap node during initial configuration to provide the required information to the master node (control plane). It boots by using an Ignition config file that describes how to create the cluster. The bootstrap node creates the master nodes, and master nodes creates the worker nodes. The master nodes install additional services in the form of a set of Operators. The OpenShift *bootstrap* node runs RHCOS 4.2.

5.1.4 Master node

The OpenShift Container Platform *master* is a server that performs control functions for the whole cluster environment. The master machines are the control plane. It is responsible for the creation, scheduling, and management of all objects specific to OpenShift. It includes API, controller manager, and scheduler capabilities in one OpenShift binary. It is also a common practice to install an etcd key-value store on OpenShift *masters* to achieve a low-latency link between etcd and OpenShift *masters*. It is recommended that you run both OpenShift *masters* and etcd in highly available environments. This can be achieved by running multiple OpenShift *masters* in conjunction with an external active-passive load balancer and the clustering functions of etcd. The Controller Manager Server watches etcd for changes to objects such as replication, namespace, and service account controller objects, and then uses the API to enforce the specified state. Several such processes create a cluster with one active leader at a time. The OpenShift *master* node runs either RHCOS4.2

5.1.5 Worker node

The OpenShift *worker* nodes run containerized applications created and deployed by developers. An OpenShift *worker* node contains the OpenShift node components, including the container engine CRI-O, container workloads running and stopping executor Kubelet, and a service proxy managing across worker nodes communication for pods. An OpenShift *application* node runs RHCOS4.2 or RHEL7.6.

5.2 OpenShift components architecture

Kubernetes is an open source project developed by Google. The project gained popularity via its open and flexible architecture for managing containerized workloads at large scale. It provides APIs that can be easily integrated into other tools such as the Red Hat OpenShift Container platform. Kubernetes provides the orchestration capabilities for containers, including scheduling the container images to nodes in a cluster, managing the container life cycle, availability, replication, persistent and non-persistent storage for containers, policy, multi-tenancy, network virtualization, routing, hierarchical clusters via federation APIs, and so forth.

A software description of the Kubernetes components is described on this website: docs.openshift.com/container-platform/4.2/architecture/architecture.html.

Figure 6 shows the OpenShift high-level architecture and components.

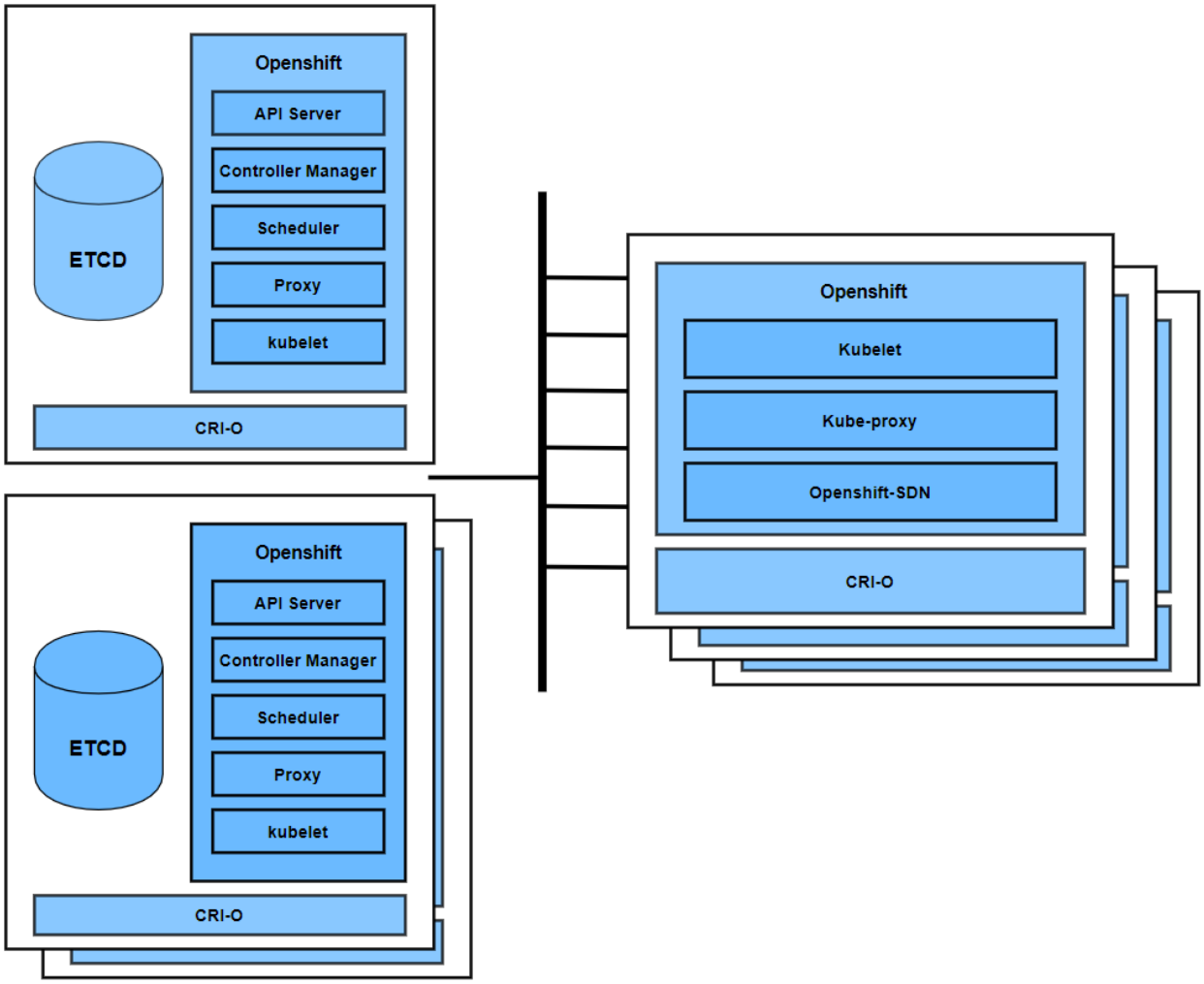

Figure 6. OpenShift component architecture

The master nodes, as described previously, are responsible for core services such as API interface, authentication/authorization, container scheduling, controller management, and configuration database. The master manages the state of the cluster and the lifecycle of the user container images. For redundancy and high availability, you can have multiple master nodes with frontend load-balancers such as HAproxy. The command line interface to the master nodes is implemented via the “oc” command.

The worker nodes are where users’ container images are executed. In OpenShift terminology the worker nodes run “pods”, each of which manages one or more running containers. Each node implements a “kubelet”, which is the node level controller that manages the pods and interacts with the OpenShift master.

In addition to the core OpenShift services, the Red Hat OpenShift platform also includes other features such as the Web based user self-service console, monitoring, an integrated container registry, storage management, authentication/authorization, automation via operators, and other administrative tools for managing the container platform.

6 Operational model

This chapter describes the options for mapping the logical components of Red Hat OpenShift onto Lenovo ThinkSystem servers, storage, and Lenovo network switches.

6.1 Hardware components

The following section describes the hardware components that can be used in an OpenShift implementation.

6.1.1 Lenovo ThinkSystem SR630 1U Server

Lenovo ThinkSystem SR630 is a 2-socket 1U rack server for enterprises that need industry-leading reliability, management, and security, as well as maximizing performance and flexibility for future growth. The SR630 server is designed to handle a wide range of workloads, such as databases, virtualization and cloud computing, virtual desktop infrastructure (VDI), infrastructure security, systems management, enterprise applications, collaboration/email, streaming media, web, and HPC.

- ThinkSystem SR630 supports two Intel Xeon Processor Scalable Family processors with up to 28-core processors, up to 38.5 MB of last level cache (LLC), up to 2666 MHz memory speeds, and up to 10.4 GT/s Ultra Path Interconnect (UPI) links.
- Offers flexible and scalable internal storage in a 1U rack form factor with up to 12x 2.5-inch drives for performance-optimized configurations or up to 4x 3.5-inch drives for capacity-optimized configurations, providing a wide selection of SAS/SATA HDD/SSD and PCIe NVMe SSD types and capacities.
- Provides I/O scalability with the LOM slot, PCIe 3.0 slot for an internal storage controller, and up to three PCI Express (PCIe) 3.0 I/O expansion slots in a 1U rack form factor.

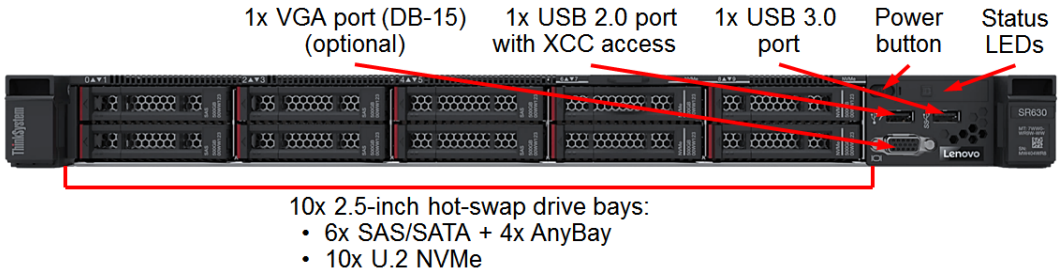

Figure 7. Lenovo ThinkSystem SR630 Server front and rear views

More detailed information, see: [ThinkSystem SR630 Server Product Guide](#).

6.1.2 Lenovo ThinkSystem SR650 2U Server

The Lenovo ThinkSystem SR650 is an ideal 2-socket 2U rack server for small businesses up to large enterprises that need industry-leading reliability, management, and security, as well as maximizing performance and flexibility for future growth. The SR650 server is particularly suited for big data applications due to its rich internal data storage, large internal memory and selection of high performance Intel processors. It is also designed to handle general workloads, such as databases, virtualization and cloud computing, virtual desktop infrastructure (VDI), enterprise applications, collaboration/email, and business analytics.

The SR650 server supports:

- Up to two Intel® Xeon® Scalable Processors
- Up to 1.5 TB 2666 MHz TruDDR4 memory (support for up to 3 TB is planned for future),
- Up to 24x 2.5-inch or 14x 3.5-inch drive bays with an extensive choice of NVMe PCIe SSDs, SAS/SATA SSDs, and SAS/SATA HDDs
- Flexible I/O Network expansion options with the LOM slot, the dedicated storage controller slot, and up to 6x PCIe slots

Figure 8. Lenovo ThinkSystem SR650

Combined with the Intel® Xeon® Scalable Processors (Bronze, Silver, Gold, and Platinum), the Lenovo SR650 server offers an even higher density of workloads and performance that lowers the total cost of ownership (TCO). Its pay-as-you-grow flexible design and great expansion capabilities solidify dependability for any kind of workload with minimal downtime.

The SR650 server provides high internal storage density in a 2U form factor with its impressive array of workload-optimized storage configurations. It also offers easy management and saves floor space and power consumption for most demanding use cases by consolidating storage and server into one system.

This reference architecture recommends the storage-rich ThinkSystem SR650 for the following reasons:

- **Storage capacity**: The nodes are storage-rich. Each of the 14 configured 3.5-inch drives has raw capacity up to 10 TB and each, providing for 140 TB of raw storage per node and over 2000 TB per rack.
- **Performance**: This hardware supports the latest Intel® Xeon® Scalable processors and TruDDR4 Memory.
- **Flexibility**: Server hardware uses embedded storage, which results in simple scalability (by adding nodes).
- **PCIe slots**: Up to 7 PCIe slots are available if rear disks are not used, and up to 3 PCIe slots if the Rear HDD kit is used. They can be used for network adapter redundancy and increased network throughput.
- **Higher power efficiency**: Titanium and Platinum redundant power supplies that can deliver 96% (Titanium) or 94% (Platinum) efficiency at 50% load.
- **Reliability:** Outstanding reliability, availability, and serviceability (RAS) improve the business environment and helps save operational costs

For more information, see the Lenovo ThinkSystem SR650 Product Guide:

https://lenovopress.com/lp0644-lenovo-thinksystem-sr650-server

6.1.3 Lenovo ThinkSystem NE1032/NE1032T Rack Switch

The Lenovo ThinkSystem NE1032/NE1032T RackSwitch family is a 1U rack-mount 10 Gb Ethernet switch that delivers lossless, low-latency performance with feature-rich design that supports virtualization, Converged Enhanced Ethernet (CEE), high availability, and enterprise class Layer 2 and Layer 3 functionality. The hot-swap redundant power supplies and fans (along with numerous high-availability features) help provide high availability for business sensitive traffic. These switches deliver line-rate, high-bandwidth switching, filtering, and traffic queuing without delaying data.

The NE1032 RackSwitch has 32x SFP+ ports that support 1 GbE and 10 GbE optical transceivers, active optical cables (AOCs), and direct attach copper (DAC) cables.

Figure 9. Lenovo ThinkSystem NE1032 RackSwitch

For more information, see the ThinkSystem NE1032 Product Guide.

The NE1032T RackSwitch has 24x 1/10 Gb Ethernet (RJ-45) fixed ports and 8x SFP+ ports that support 1 GbE and 10 GbE optical transceivers, active optical cables (AOCs), and direct attach copper (DAC) cables.

Figure 10. Lenovo ThinkSystem NE1032T RackSwitch

For more information, see the ThinkSystem NE1032T Product Guide.

6.1.4 Lenovo ThinkSystem NE2572 RackSwitch

For scale-out OpenShift Container Platform implementations to support 1000s of container images with high-performance requirements for network as well as storage, it is recommended to use 25Gbps Ethernet networking as the fabric.

The Lenovo ThinkSystem NE2572 RackSwitch is designed for the data center and provides 10 Gb/25 Gb Ethernet connectivity with 40 Gb/100 Gb Ethernet upstream links. It is ideal for big data, cloud, and enterprise workload solutions. It is an enterprise class Layer 2 and Layer 3 full featured switch that delivers line-rate, high-bandwidth switching, filtering, and traffic queuing without delaying data.

The NE2572 RackSwitch has 48x SFP28/SFP+ ports that support 10 GbE SFP+ and 25 GbE SFP28 optical transceivers, active optical cables (AOCs), and direct attach copper (DAC) cables. The switch also offers 6x QSFP28/QSFP+ ports that support 40 GbE QSFP+ and 100 GbE QSFP28 optical transceivers, active optical cables (AOCs), and direct attach copper (DAC) cables. The QSFP28/QSFP+ ports can also be split out into

two 50 GbE (for 100 GbE QSFP28), or four 10 GbE (for 40 GbE QSFP+) or 25 GbE (for 100 GbE QSFP28) connections by using breakout cables.

Figure 11. Lenovo ThinkSystem NE2572 Rack Switch

For more information, see the Lenovo ThinkSystem NE2572 Switch Product Guide.

6.1.5 Lenovo RackSwitch G8052

The Lenovo System Networking RackSwitch G8052 (as shown in Figure 12) is an Ethernet switch that is designed for the data center and provides a virtualized, cooler, and simpler network solution. The Lenovo RackSwitch G8052 offers up to 48 1 GbE ports and up to 4 10 GbE ports in a 1U footprint. The G8052 switch is always available for business-sensitive traffic by using redundant power supplies, fans, and numerous high-availability features. For more information, see this website: lenovopress.com/tips1270.

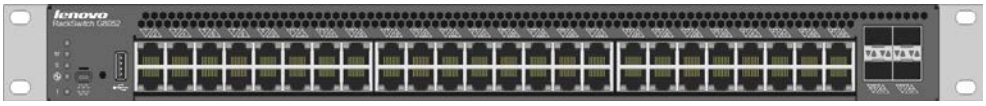

Figure 12. Lenovo RackSwitch G8052

6.2 Deployment models

The OpenShift Container Platform can be implemented in development/test, staging, and production settings. Each node role has its own dedicated servers for performance and availability. However, in a non-production environment, a minimal environment can be provided to test applications before moving them to a staging or production environment.

For a production OpenShift deployment, all of the core services such as the API servers, Kubernetes scheduler, etcd, etc., need to be highly available. The table below shows the recommended configuration for a production deployment.

Node type	Quantity	Node role
Bastion	1	Deployment of the environment, Ansible playbooks, hardware management, etc.
Load Balance	2	HAProxy, Keepalive, routing, etcd, logging, metrics.
Bootstrap	1	OpenShift bootstrap node. It can be a VM.
Master (Control plane)	3	OpenShift API master, Kubernetes scheduler
Worker	2+	Runs the application containers
Storage	1+	Storage as OpenShift platform's backend storage

There are performance and availability implications of running the Red Hat OpenShift Container Storage alongside the workload containers in a hyperconverged environment. For production environments, it is recommended to separate hyper-converged compute servers from storage-only servers, or to ensure that the servers have sufficient CPU, memory, and storage resources to avoid any performance bottlenecks.

6.3 Auto-Deployment by Lenovo Open Cloud - Automation

Lenovo Open Cloud - Automation (LOC-A) are Lenovo next generation software stack for simplifying Cloud and Data Center Infrastructure Deployment & Management. LOC-A leverages open software solutions to support Red Hat (LOC Automation for RH) and VMWare (LOC Automation for VCF) based flavors. LOC-A provides RedHat OpenShift Container Platform one-click auto-deployment from scratch by following pre-defined HW configurations and pre-defined deployment workflow.

Figure 13 shows the open software solutions components in LOC-A.

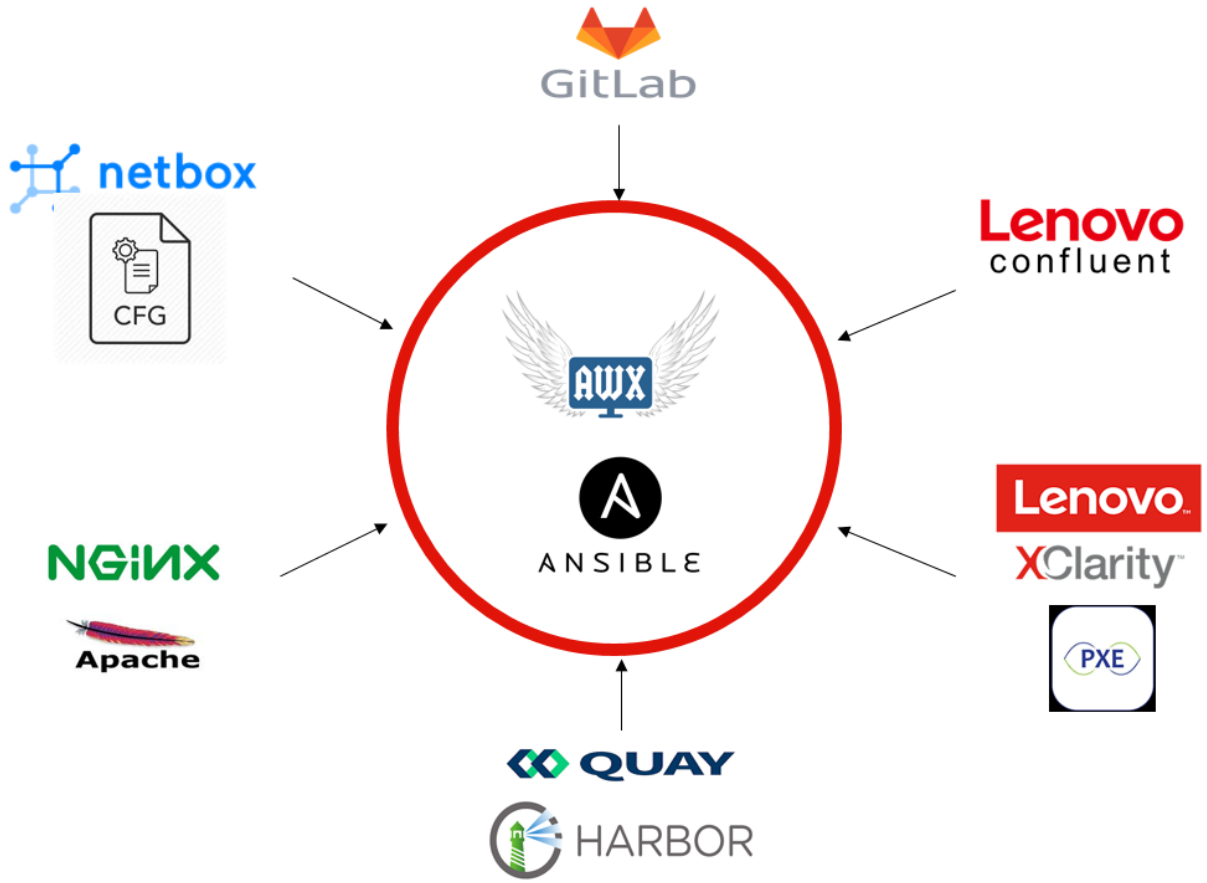

Figure 13. Lenovo Open Cloud-Automation Components

LOC-A uses ansible and AWS tool to perform auto-deployment of multiple platforms. Platform configurations/resources are stored in Netbox tool or configuration files. Lenovo confluent are a discovery engine to find resources on site. Lenovo XClarity and PXE servers are used for Hardware management, firmware upgrade, and OS deployment. Nginx/Apache server manage VM images. Quay/Harbor manage container images.

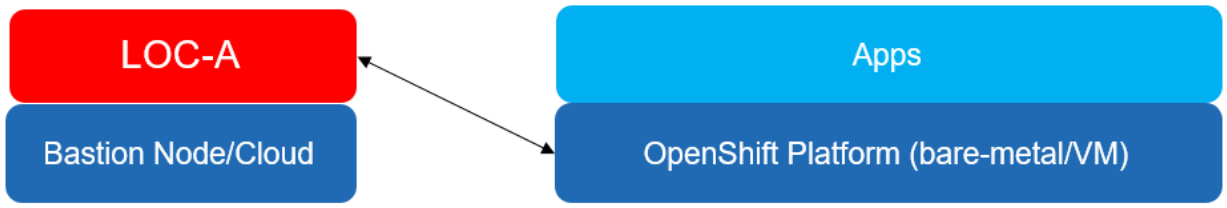

Figure 14. OpenShift Deployed by Lenovo Open Cloud - Automation

LOC-A can be installed on bastion node/cloud. It deploys OpenShift platform on top of bare-metal HW or virtualization platform.

6.4 Compute servers

The OpenShift Container Platform can be implemented on a small footprint of x86 servers clustered together and scaled as the user workloads grow.

The right choice of servers and the corresponding hardware configuration for CPUs, memory, and networking will depend upon various factors, including but not limited to:

- Number of concurrent OpenShift users to be supported
- Type and mix of application workloads, which will drive the system resource requirements
- System growth projection
- Development or production use
- Fault-tolerance and availability requirements for applications
- Application performance expectations
- Implementation of hybrid-cloud model, which drives the requirements for on-premises infrastructure

For more guidance on sizing and other considerations is available for OpenShift clusters in OpenShift documentation: access.redhat.com/documentation/en-us/openshift_container_platform/3.9/html-single/scaling_and_performance_guide/#scaling-performance-cluster-limits

Lenovo does not recommend server configuration specifics for CPU, memory, storage, etc. because it is heavily dependent on the sizing considerations previously listed. However, Lenovo has verified configurations on ThinkSystem servers using both Intel Xeon Scalable Processors gen 1 and gen 2 CPUs. The user is requested to perform a proper sizing assessment for their particular needs and choose the hardware configurations to meet those requirements.

6.5 Persistent storage for containerized workloads

There are two types of storage consumed by containerized applications – ephemeral (non-persistent) and persistent. As the names suggest, non-persistent storage is created and destroyed along with the container and is only used by applications during their lifetime as a container. Hence, non-persistent storage is used for

temporary data. When implementing the OpenShift Container Platform, local disk space on the application nodes can be configured and used for the non-persistent storage volumes.

Persistent storage, on the other hand, is used for data that needs to be persisted across container instantiations. An example is a 2 or 3-tier application that has separate containers for the web and business logic tier and the database tier. The web and business logic tier can be scaled out using multiple containers for high availability. The database that is used in the database tier requires persistent storage that is not destroyed.

OpenShift uses a persistent volume framework that operates on two concepts – persistent storage and persistent volume claim. Persistent storage is the physical storage volumes that are created and managed by the OpenShift cluster administrator. When an application container requires persistent storage, it would create a persistent volume claim (PVC). The PVC is a unique pointer/handle to a persistent volume on the physical storage, except that PVC is not bound to a physical volume. When a container makes a PVC request, OpenShift would allocate the physical disk and binds it to the PVC. When the container image is destroyed, the volume bound to the PVC is not destroyed unless you explicitly destroy that volume. In addition, during the lifecycle of the container if it relocates to another physical server in the cluster, the PVC binding will still be maintained. After the container image is destroyed, the PVC is released, but the persisted storage volume is not deleted. The specific persistent storage policy for the volume will determine when the volume gets deleted.

For more detailed conceptual information on persistent volumes see: access.redhat.com/documentation/en-us/openshift_container_platform/4.2/html/storage

A variety of persistent storage options are available for OpenShift, choices including CSI (Container Storage Interface), OCS (Red Hat OpenShift Container Storage), NFS, Cinder, iSCSI, Azure File, AWS elastic block storage (EBS), and others. For a complete list of these choices and the corresponding requirements, see the link below: access.redhat.com/documentation/en-us/openshift_container_platform/4.2/html/storage/understanding-persistent-storage#persistent-storage-overview_understanding-persistent-storage

6.5.1 Container Storage Interface (CSI)

The Container Storage Interface (CSI) is a standard for exposing arbitrary block and file storage systems to containerized workloads on Container Orchestration Systems like OpenShift. Using CSI third-party storage providers can write and deploy plugins exposing new storage systems in OpenShift platform. OpenShift Container Platform can leverage CSI to consume storage from storage backends as persistent storage.

OpenShift platform support CSI 1.1, such as Ceph, NetApp Trident, Cinder, etc. More CSI drivers can be found in: kubernetes-csi.github.io/docs/drivers.html. NetApp Trident provides a persistent volume plugin supports ONTAP. Lenovo ThinkSystem DE and ThinkSystem DM uses ONTAP software with Lenovo servers.

For more information on Container Storage Interface, see: access.redhat.com/documentation/en-us/openshift_container_platform/4.2/html/storage/configuring-persistent-storage#persistent-storage-using-csi

Figure 15 provides a high-level overview about the Container Storage Interface components running in pods in the OpenShift Container Platform cluster. CSI driver needs its own external controllers' deployment and DaemonSet with the driver and CSI registrar

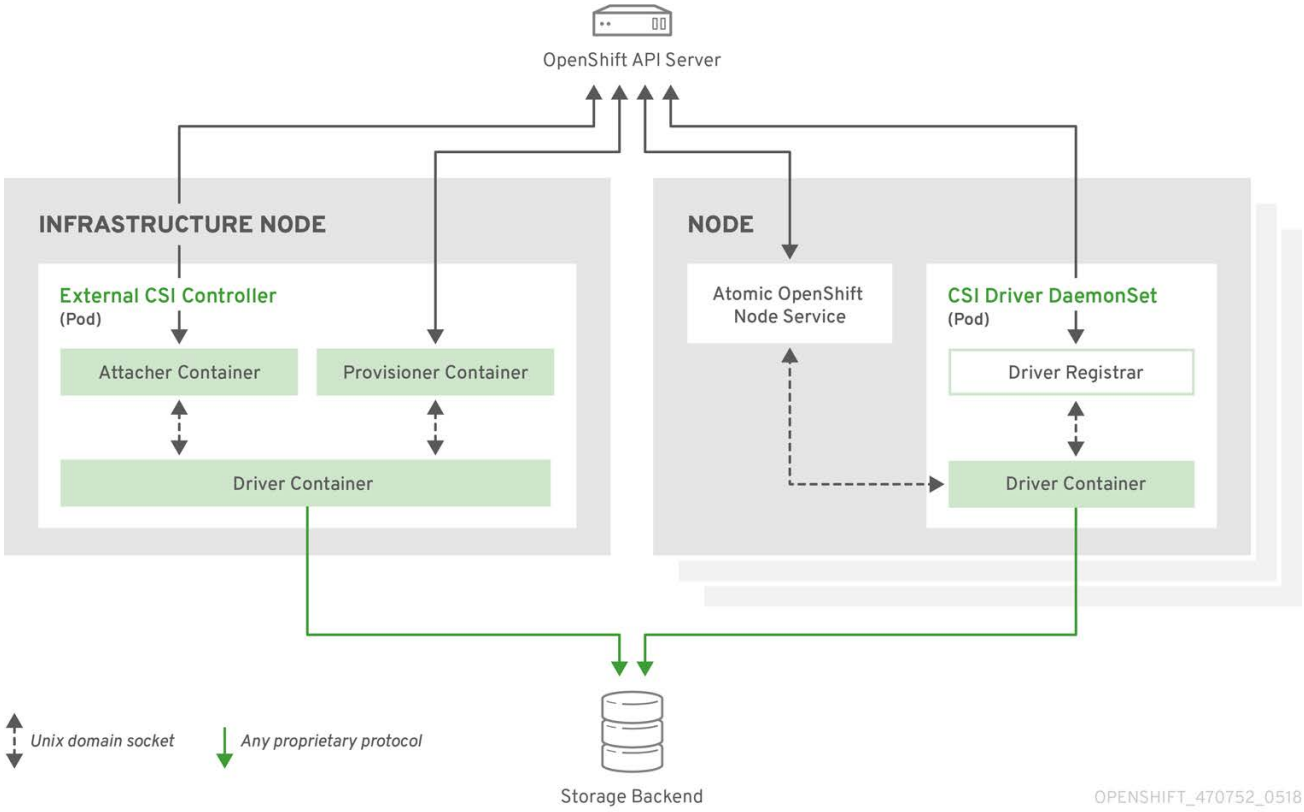

Figure 15. Container Storage Interface Architecture

6.5.2 Red Hat OpenShift Container Storage (OCS)

Red Hat OpenShift Container Storage is used as the persistent storage backend in this Reference Architecture as it simplifies the overall OpenShift architecture and consolidates the compute and storage components in the same x86 servers.

Red Hat OpenShift Container Storage is an open source distributed, scalable, and high-performance file based storage system. It is used widely for many types of applications. Red Hat OpenShift Container Storage provides volume plug-ins into OpenShift to support the persistent storage for containers.

Red Hat OpenShift Container Storage can be implemented for the OpenShift platform in two ways – *converged mode* or standalone *independent mode***.**

In the *independent* mode, it runs on its own standalone cluster and the OpenShift nodes access the storage via the persistent volume mapping. In this mode, storage is separate to the compute nodes. The storage can be scaled independently of the compute nodes by adding more servers later to the storage cluster.

For more information on Red Hat OpenShift Container Storage, see: access.redhat.com/documentation/en-us/red_hat_openshift_container_storage/4.2/html/planning_your_deployment/introduction-to-openshift-container-storage-4.2_rhocs

Figure 16 gives a high-level overview of OpenShift Container Storage architecture.

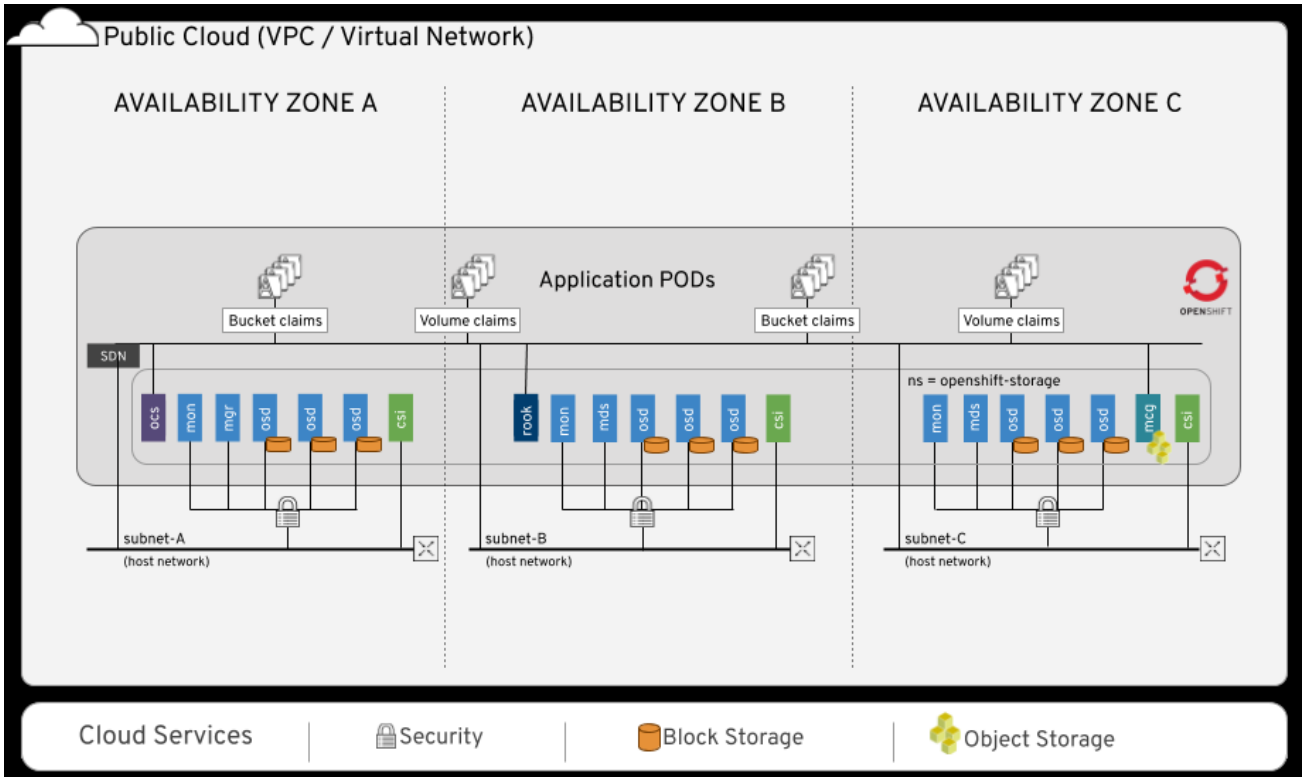

Figure 16. OpenShift Container Storage Architecture

6.6 Networking

For OpenShift Container Platform deployment, 10Gbps networking is recommended as the choice for all cluster-wide communication for the core OpenShift services, virtual network implementation for container workloads, storage services access, as well as all east-west traffic across the container workloads. In addition, the north-south traffic between the OpenShift environment and uplink into the customer (or campus) network can be implemented over the 10Gbps network.

There are three logical networks defined in this RA:

- **External**: The external network is used for the public API, the OpenShift web interface, and exposed applications (services and routes).
- **Internal**: This is the primary, non-routable network used for cluster management and inter-node communication. The same network acts as the layer for server provisioning using PXE and HTTP. Domain Name Servers (DNS) and Dynamic Host Configuration Protocol (DHCP) services also reside on this network to provide the functionality necessary for the deployment process and the cluster to work. Communication with the Internet is provided by NAT configured on the *bastion* node.
- **Out-of-band/IPMI**: This is a secured and isolated network used for switch and server hardware management, such as access to the IMM module and SoL (Serial-over-LAN).

Figure 17 shows the Red Hat OpenShift servers and the recommended network architecture.

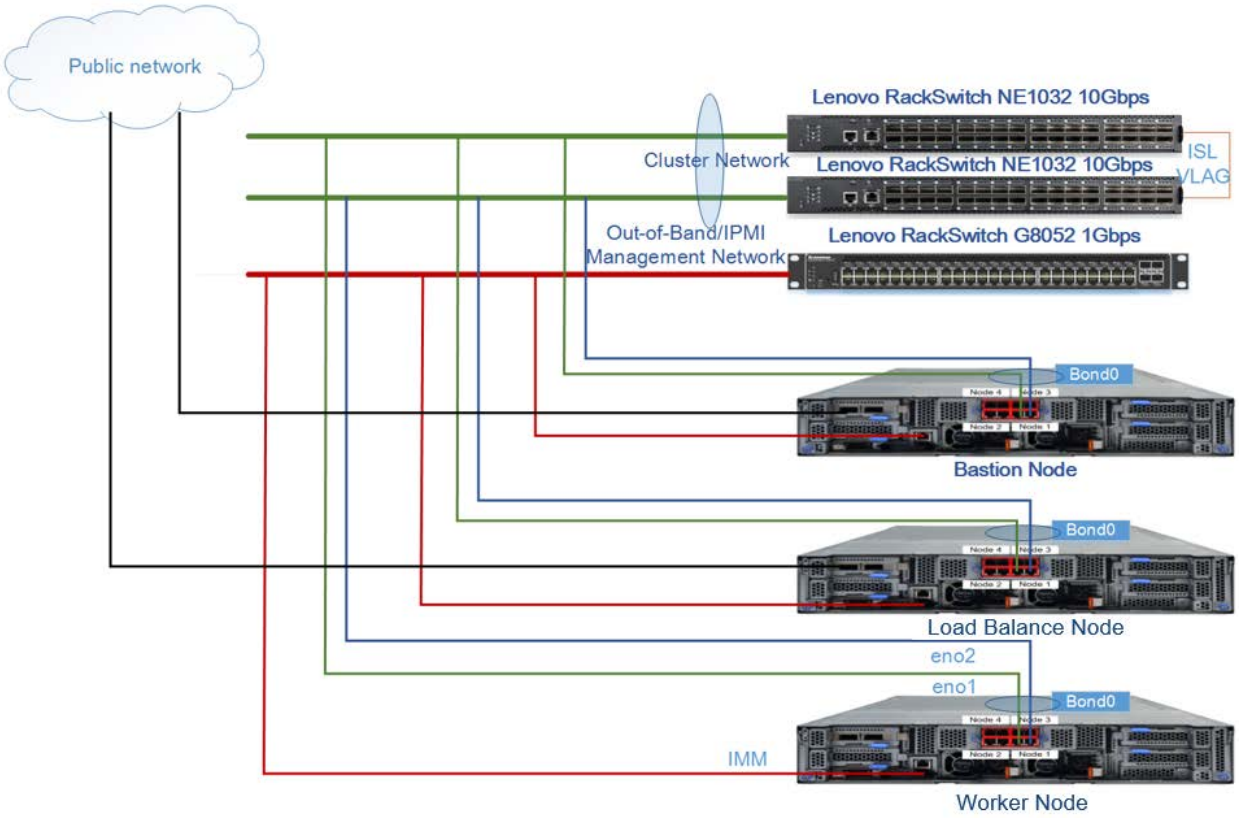

Figure 17. OpenShift Network Connectivity

All OpenShift nodes are connected via the internal network, where they can communicate with each other. Furthermore, Open vSwitch creates its own network for OpenShift pod-to-pod communication. Because of the multi-tenant plugin, Open vSwitch pods can communicate to each other only if they share the same project namespace. There is a virtual IP address managed by Keepalived on two *infrastructure* hosts for external access to the OpenShift web console and applications.

6.6.1 Hardware management network

For out-of-band management of the servers and initial cluster deployment over the network from the *bastion* node, use the 1Gbps management fabric via the Lenovo RackSwitch G8052. The Lenovo ThinkSystem rack servers have a dedicated 1GbE network port for the XCC interface. The XCC enables remote-management capabilities for the servers, access to the server's remote console for troubleshooting, and running the IPMI commands via the embedded baseboard management controller (BMC) module.

6.6.2 Network redundancy

The Lenovo OpenShift platform uses the 10 GbE network as the primary fabric for inter-node communication. Two Lenovo ThinkSystem NE1032 RackSwitch switches are used to provide redundant data layer communication and deliver maximum availability.

Figure 18 shows the redundant network architecture.

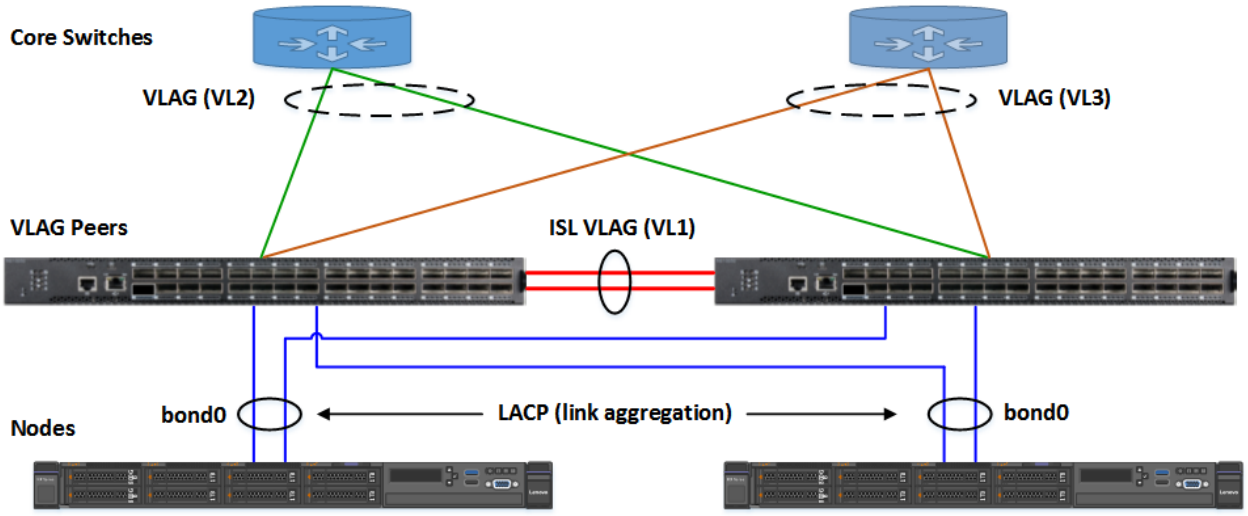

Figure 18. Redundant network architecture

Virtual Link Aggregation Group (VLAG) is a feature of the Lenovo CNOS operating system that allows a pair of Lenovo switches to work as a single virtual switch. Each of the cluster nodes has a link to each VLAG peer switch for redundancy. This provides improved high availability (HA) for the nodes using the link aggregation control protocol (LACP) for aggregated bandwidth capacity. Connection to the uplink core network is facilitated by the VLAG peers, which present a logical switch to the uplink network, enabling connectivity with all links active and without a hard requirement for spanning-tree protocol (STP). The link between the two VLAG peers is an inter-switch link (ISL) and provides excellent support of east-west cluster traffic the nodes. The VLAG presents a flexible basis for interconnecting to the uplink/core network, ensures the active usage of all available links, and provides high availability in case of a switch failure or a required maintenance outage.

6.7 Systems management

In addition to in-band management via IPMI, the Lenovo XClarity Administrator software provides centralized resource management that reduces complexity, speeds up response, and enhances the availability of Lenovo® server systems and solutions.

The Lenovo XClarity Administrator provides agent-free hardware management for Lenovo's ThinkSystem® rack servers, System x® rack servers, and Flex System™ compute nodes and components, including the Chassis Management Module (CMM) and Flex System I/O modules. Figure 19 shows the Lenovo XClarity administrator interface, in which Flex System components and rack servers are managed and are seen on the dashboard. Lenovo XClarity Administrator is a virtual appliance that is quickly imported into a virtualized environment server configuration.

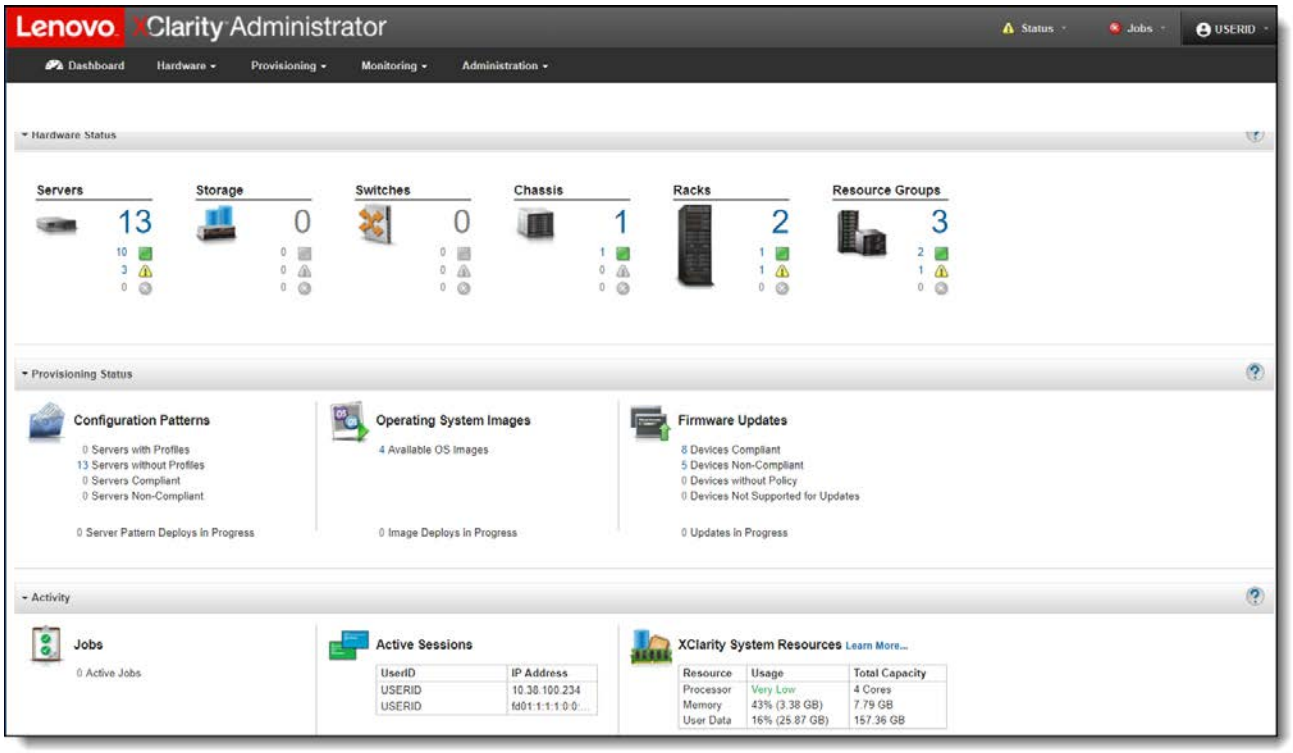

Figure 19. Lenovo XClarity Administrator Dashboard

For more information, see: Lenovo XClarity Administrator Product Guide

6.8 Deployment examples

This section describes one example configurations that have been tested by Lenovo:

- Typical OpenShift configuration

6.8.1 Typical OpenShift configuration

The typical OpenShift 4.2 configuration uses 9 nodes, 3 switches, and external storage as follows:

- 1 Bastion node
- 2 Load Balance nodes
- 1 Bootstrap node (Bootstrap node is a temporary node that can be removed after deployment)
- 3 Master nodes
- 2 worker nodes
- 2 10GbE switch for traffic load network
- 1 1GbE switch for management network
- storage provide a storage cluster as Openshift backend storage.

This configuration represents a production grade OpenShift implementation that meets high-availability, redundancy, and scale requirements for enterprises. Additional Application nodes can be added to increase the available compute and storage capacity.

Table 3 provides the hardware configuration summary using Lenovo ThinkSystem SR630 servers. The detailed server BOMs can be found in “Appendix A: Lenovo bill of materials” on page 30.

Table 3. Node Hardware Configuration for typical OpenShift Deployment

OpenShift Node Role	ThinkSystem SR630 server configuration
Bastion Node Master Node Bootstrap Node Worker Node	2x Intel Xeon Gold 6126 12C 125W 2.6GHz Processor 384GB memory (12x 32 GB) 2x ThinkSystem M.2 5300 480GB SATA 6Gbps Non-Hot Swap SSD 1x ThinkSystem M.2 with Mirroring Enablement Kit 1x ThinkSystem 430-8i SAS/SATA 12Gb Dense HBA 10x ThinkSystem 2.5" 5300 3.84TB Entry SATA 6Gb Hot Swap SSD 1x ThinkSystem 10Gb 4-port SFP+ LOM
Enhanced Worker node	2x Intel Xeon Gold 6140 18C 140W 2.3GHz Processor 384GB memory (12x 32 GB) 2x ThinkSystem M.2 5300 480GB SATA 6Gbps Non-Hot Swap SSD 1x ThinkSystem M.2 with Mirroring Enablement Kit 10x ThinkSystem 2.5" 5300 3.84TB Entry SATA 6Gb Hot Swap SSD 1x ThinkSystem 430-8i SAS/SATA 12Gb Dense HBA 10x ThinkSystem 2.5" 5300 3.84TB Entry SATA 6Gb Hot Swap SSD 1x ThinkSystem 10Gb 4-port SFP+ LOM

The server configuration for the Bastion, Master and Worker nodes is the same. This allows the role for a server to be easily changed. The configuration for all of the Application nodes is also the same, regardless of whether a particular server is used for hyper-converged compute or storage only.

6.9 Software

For this example, the following software is needed:

- **OpenShift Container Platform**, which adds developer- and operation-centric tools to enable rapid application development, easy deployment, scaling, and long-term lifecycle maintenance for small and large teams and applications
- **Lenovo XClarity Administrator** for management of the operating systems on bare-metal servers

In addition, the OpenShift Container Platform requires the following software packages:

- **Kubernetes** to orchestrate and manage containerized applications
- **Etcd***, which is a key-value store for the OpenShift Container Platform cluster
- **Open vSwitch*** to provide software-defined networking (SDN)-specific functions in the OpenShift Container Platform environment
- **HAProxy*** for routing and load-balancing purposes
- **Keepalived*** for virtual IP management for HAProxy instances

Table 4 lists the software versions used for this example deployment

Table 4. Software versions

Component	Version
Red Hat Enterprise Linux	8.1
Red Hat CoreOS	4.2
OpenShift Container Platform	4.2

6.10 Deployment validation

The deployment should be validated before it is used. The OpenShift Container Platform web console provides two perspectives; the Administrator perspective and the Developer perspective. At verification step, log on to the OpenShift Container Platform web console using the following URL address: https://openshift.ocp.example.com:8443 and display the OpenShift container Administrator perspective. Figure 20 shows an example.

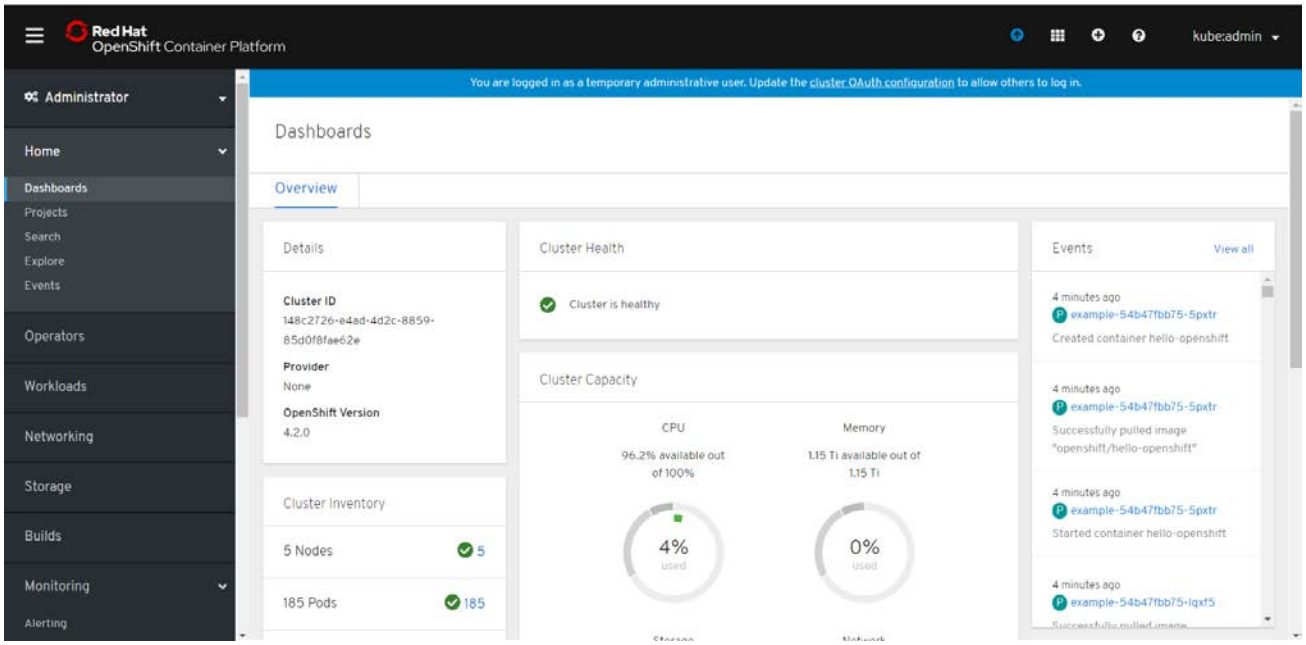

Figure 20. Administrator perspective of OpenShift Container Platform

Figure 21 shows Developer perspective.

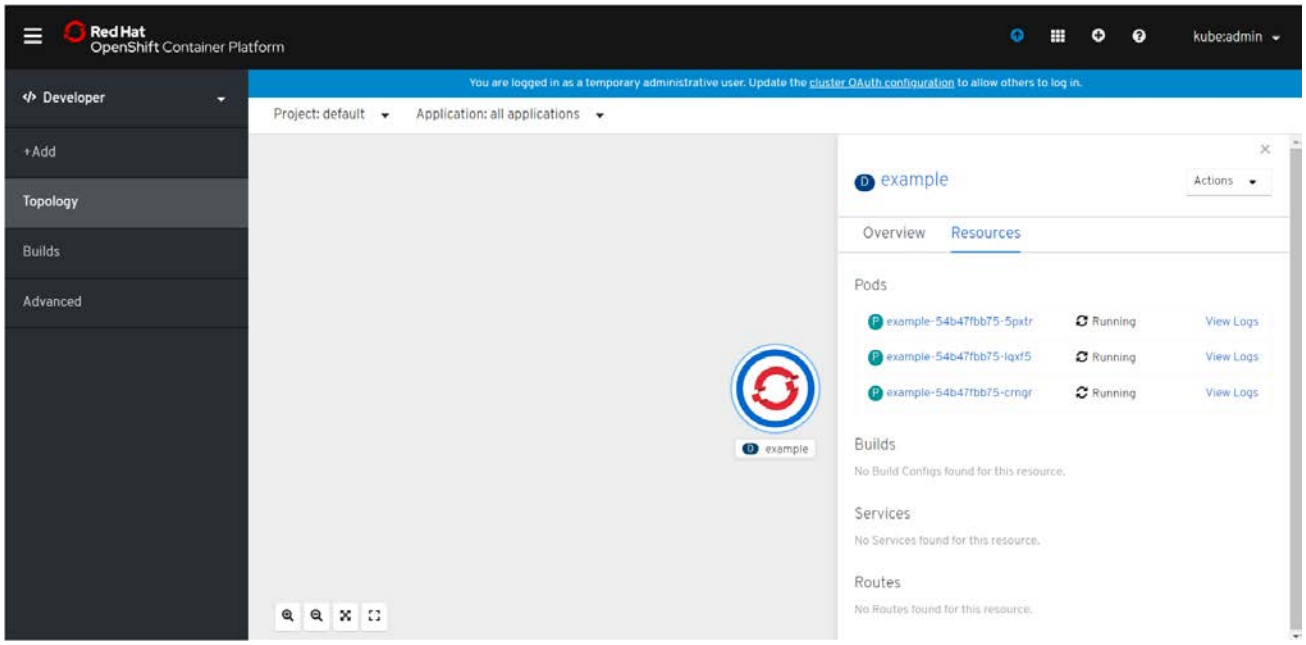

Figure 21. Developer perspective of OpenShift Container Platform

7 Appendix A: Lenovo bill of materials

This appendix contains the bill of materials (BOMs) for different configurations of hardware for OpenShift deployments. There are sections for servers and networking. For OpenShift, some servers with the same configurations are deployed as different types of nodes.

7.1 Server BOM for typical configuration

This section contains the server BOMs for a typical configuration.

Lenovo ThinkSystem SR630 (Bastion Node, LoadBalance Node, Master Node, Worker Node)

Code	Description	Quantity
7X02CTO1WW	Server : ThinkSystem SR630 - 3yr Warranty	1
AUW0	ThinkSystem SR630 2.5" Chassis with 8 Bays	1
AWEL	Intel Xeon Gold 6126 12C 125W 2.6GHz Processor	2
AUND	ThinkSystem 32GB TruDDR4 2666 MHz (2Rx4 1.2V) RDIMM	12
AUWB	ThinkSystem SR530/SR630/SR570 2.5" SATA/SAS 8-Bay Backplane	1
5977	Select Storage devices - no configured RAID required	1
AUNL	ThinkSystem 430-8i SAS/SATA 12Gb HBA	1
B8JP	ThinkSystem 2.5" 5300 3.84TB Entry SATA 6Gb Hot Swap SSD	4
AUMV	ThinkSystem M.2 with Mirroring Enablement Kit	1
B919	ThinkSystem M.2 5300 480GB SATA 6Gbps Non-Hot Swap SSD	2
AUKK	ThinkSystem 10Gb 4-port SFP+ LOM	1
AVWB	ThinkSystem 1100W (230V/115V) Platinum Hot-Swap Power Supply	2
6400	2.8m, 13A/100-250V, C13 to C14 Jumper Cord	2
AUPW	ThinkSystem XClarity Controller Standard to Enterprise Upgrade	1
AXCA	ThinkSystem Toolless Slide Rail	1
A1PJ	3m Passive DAC SFP+ Cable	2
A51P	2m Passive DAC SFP+ Cable	2
B0MJ	Feature Enable TPM 1.2	1
B7XZ	Disable IPMI-over-LAN	1
AWGE	ThinkSystem SR630 WW Lenovo LPK	1
AUWX	8x2.5" HDD BP Cable Kit	1
AURS	Lenovo ThinkSystem Memory Dummy	12
AUTC	ThinkSystem SR630 Lenovo Agency Label	1
AVEN	ThinkSystem 1x1 2.5" HDD Filler	4
AUW3	Lenovo ThinkSystem Mainstream MB - 1U	1
AUW7	ThinkSystem SR630 4056 Fan Module	2
AULP	ThinkSystem 1U CPU Heatsink	2
AVJ2	ThinkSystem 4R CPU HS Clip	2
AUTJ	ThinkSystem common Intel Label	1
AUTA	XCC Network Access Label	1
AUTV	ThinkSystem large Label for non-24x2.5"/12x3.5"/10x2.5"	1

AVWK	ThinkSystem EIA Plate with Lenovo Logo	1
AUX4	MS 1U Service Label LI	1
AWF9	ThinkSystem Response time Service Label LI	1
AUX3	ThinkSystem SR630 Model Number Label	1
AUX0	ThinkSystem Package for SR630	1
AUT8	ThinkSystem 1100W RDN PSU Caution Label	1
AUWM	Lenovo ThinkSystem 1U LP+LP BF Riser Dummy	1
AUWL	Lenovo ThinkSystem 1U LP Riser Dummy	1
AUWF	Lenovo ThinkSystem Super Cap Holder Dummy	1
B173	Companion Part for XClarity Controller Standard to Enterprise Upgrade in Factory	1
AUWG	Lenovo ThinkSystem 1U VGA Filler	1
B0ML	Feature Enable TPM on MB	1
5PS7A01504	Essential Service - 3Yr 24x7 4Hr Resp + YDYD SXM SR630	1
5AS7A02045	Hardware Installation Server (Business Hours)	1
5641PX3	XClarity Pro, Per Endpoint w/3 Yr SW S&S	1
1340	Lenovo XClarity Pro, Per Managed Endpoint w/3 Yr SW S&S	1
3444	Serial Number Only	1

Lenovo ThinkSystem SR650 (Enhanced Worker Node)

Code	Description	Quantity
7X06CTO1WW	Server : ThinkSystem SR650 - 3yr Warranty	1
AUVV	ThinkSystem SR650 2.5" Chassis with 8, 16 or 24 bays	1
AWE1	Intel Xeon Gold 6140 18C 140W 2.3GHz Processor	2
AUND	ThinkSystem 32GB TruDDR4 2666 MHz (2Rx4 1.2V) RDIMM	12
AURA	ThinkSystem 2U/Twr 2.5" SATA/SAS 8-Bay Backplane	2
5977	Select Storage devices - no configured RAID required	1
AUNM	ThinkSystem 430-16i SAS/SATA 12Gb HBA	1
B8JP	ThinkSystem 2.5" 5300 3.84TB Entry SATA 6Gb Hot Swap SSD	10
AUKK	ThinkSystem 10Gb 4-port SFP+ LOM	1
AVWF	ThinkSystem 1100W (230V/115V) Platinum Hot-Swap Power Supply	2
6400	2.8m, 13A/100-250V, C13 to C14 Jumper Cord	2
AUPW	ThinkSystem XClarity Controller Standard to Enterprise Upgrade	1
AXCA	ThinkSystem Toolless Slide Rail	1
AURD	ThinkSystem 2U left EIA Latch Standard	1
A51P	2m Passive DAC SFP+ Cable	2
A1PJ	3m Passive DAC SFP+ Cable	2
B0MJ	Feature Enable TPM 1.2	1

B7XZ	Disable IPMI-over-LAN	1
AUQB	Lenovo ThinkSystem Mainstream MB - 2U	1
AUSH	MS First 2U 8x2.5" HDD BP Cable Kit	1
AUSM	MS 2nd 2U 8X2.5" Cable Kit	1
B0ML	Feature Enable TPM on MB	1
AWFF	ThinkSystem SR650 WW Lenovo LPK	1
AVEQ	ThinkSystem 8x1 2.5" HDD Filler	1
AVEN	ThinkSystem 1x1 2.5" HDD Filler	6
AUTA	XCC Network Access Label	1
AVJ2	ThinkSystem 4R CPU HS Clip	2
AUSF	Lenovo ThinkSystem 2U MS CPU Performance Heatsink	2
AUSG	ThinkSystem SR650 6038 Fan module	1
B173	Companion Part for XClarity Controller Standard to Enterprise Upgrade in Factory	1
AUTJ	ThinkSystem common Intel Label	1
AURS	Lenovo ThinkSystem Memory Dummy	12
AUTQ	ThinkSystem small Lenovo Label for 24x2.5"/12x3.5"/10x2.5"	1
AWF9	ThinkSystem Response time Service Label LI	1
AUSZ	ThinkSystem SR650 Service Label LI	1
AVWK	ThinkSystem EIA Plate with Lenovo Logo	1
AUTD	ThinkSystem SR650 model number Label	1
AUT8	ThinkSystem 1100W RDN PSU Caution Label	1
AURT	Lenovo ThinkSystem 2U 3FH Riser Dummy	1
AURF	Lenovo ThinkSystem 2U 2FH Riser Dummy	1
AUSA	Lenovo ThinkSystem M3.5" Screw for EIA	4
AUSU	ThinkSystem Package for SR650	1
AUT1	ThinkSystem SR650 Lenovo Agency Label	1
AUTY	ThinkSystem 12-15 sequence Label for 24x2.5"Chassis	1
AUTF	ThinkSystem 4-7 sequence Label for 2U 2.5"	1
A2HP	Configuration ID 01	1
5374CM1	Configuration Instruction	1
AVE5	ThinkSystem 430-16i SAS/SATA 12Gb HBA placement	1
A2JX	Controller 01	1
A2HP	Configuration ID 01	1
5PS7A06897	Premier Essential - 3Yr 24x7 4Hr Resp + YDYD SR650	1
5AS7A02045	Hardware Installation Server (Business Hours)	1

5641PX3	XClarity Pro, Per Endpoint w/3 Yr SW S&S	1
1340	Lenovo XClarity Pro, Per Managed Endpoint w/3 Yr SW S&S	1
3444	Serial Number Only	1

7.2 Networking BOM

This section contains the BOMs for network switches. The typical configuration can use 10 or 25 GbE networking. The Intel Select configuration must use 25 GbE networking.

Lenovo ThinkSystem NE2572 Switch

Code	Description	Quantity
7159HE3	Lenovo ThinkSystem NE2572 RackSwitch (Rear to Front)	1
AV19	Lenovo ThinkSystem NE2572 RackSwitch (Rear to Front)	1
AV1W	1m Passive 25G SFP28 DAC Cable	1
6204	2.8m, 10A/100-250V, C13 to IEC 320-C20 Rack Power Cable	2

Lenovo ThinkSystem NE1032 Switch

Code	Description	Quantity
7159HD1	Lenovo ThinkSystem NE1032 RackSwitch (Rear to Front)	1
AU3A	Lenovo ThinkSystem NE1032 RackSwitch (Rear to Front)	1
A1PH	1m Passive DAC SFP+ Cable	1
6204	2.8m, 10A/100-250V, C13 to IEC 320-C20 Rack Power Cable	2

Lenovo RackSwitch G8052

Code	Description	Quantity
7159G52	Lenovo System Networking RackSwitch G8052 (Rear to Front)	1
6201	1.5m, 10A/100-250V, C13 to IEC 320-C14 Rack Power Cable	2
3802	1.5m Blue Cat5e Cable	3
A3KP	Lenovo System Networking Adjustable 19" 4 Post Rail Kit	1

Resources

- Architecture of the Red Hat OpenShift Container Platform
  access.redhat.com/documentation/en-us/openshift_container_platform/4.2/html/architecture/architecture

- OpenShift Container Platform 4.2
  access.redhat.com/documentation/en-us/openshift_container_platform/4.2/

- Kubernetes
  kubernetes.io/ and kubernetes.io/docs/tutorials/kubernetes-basics/

- OpenShift containerized installation
  access.redhat.com/documentation/en-us/openshift_container_platform/4.2/html/installing

- Storage in OpenShift Container Platform 4.2
  access.redhat.com/documentation/en-us/openshift_container_platform/4.2/html/storage

- Lenovo ThinkSystem DE Series
  lenovo.com/us/en/data-center/storage/storage-area-network/thinksystem-de-series/c/thinksystem-de-series

- Lenovo ThinkSystem DM Series
  lenovo.com/us/en/data-center/storage/storage-area-network/thinksystem-dm-series/c/thinksystem-dm-series

Document History

Version 1.0	1 October 2018	• Initial version
Version 1.1	1 April 2019	• Updated to include the Intel Select Base configuration
Version 1.2	22 April 2019	• Updated configurations to use Intel Xeon Scalable Processor gen 2 CPUs.
Version 2.0	24 April 2020	• Update to OpenShift 4.2

Trademarks and special notices

© Copyright Lenovo 2020.

References in this document to Lenovo products or services do not imply that Lenovo intends to make them available in every country.

Lenovo, the Lenovo logo, ThinkSystem, ThinkCentre, ThinkVision, ThinkVantage, ThinkPlus and Rescue and Recovery are trademarks of Lenovo.

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel Inside (logos), MMX, and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

Nutanix is a trademark of Nutanix, Inc. in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Information is provided "AS IS" without warranty of any kind.

All customer examples described are presented as illustrations of how those customers have used Lenovo products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.

Information concerning non-Lenovo products was obtained from a supplier of these products, published announcement material, or other publicly available sources and does not constitute an endorsement of such products by Lenovo. Sources for non-Lenovo list prices and performance numbers are taken from publicly available information, including vendor announcements and vendor worldwide homepages. Lenovo has not tested these products and cannot confirm the accuracy of performance, capability, or any other claims related to non-Lenovo products. Questions on the capability of non-Lenovo products should be addressed to the supplier of those products.

All statements regarding Lenovo future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. Contact your local Lenovo office or Lenovo authorized reseller for the full text of the specific Statement of Direction.

Some information addresses anticipated future capabilities. Such information is not intended as a definitive statement of a commitment to specific levels of performance, function or delivery schedules with respect to any future products. Such commitments are only made in Lenovo product announcements. The information is presented here to communicate Lenovo’s current investment and development activities as a good faith effort to help with our customers' future planning.

Performance is based on measurements and projections using standard Lenovo benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Photographs shown are of engineering prototypes. Changes may be incorporated in production models.

Any references in this information to non-Lenovo websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this Lenovo product and use of those websites is at your own risk.